Interface
2003 June

［表紙デザイン：㈱プランニング・ロケッツ］

Contents 2003 June

特集

ネットワークの基礎から音声通話への応用まで

Interface

話題のテクノロジ解説

ショウレポート&コラム

一般解説&連載

情報のページ

連載「音楽配信技術の最新動向」,「プログラミングの要」,「やり直しのための信号数学」,「開発技術者のためのアセンブラ入門」は,お休みさせていただきます.

Contents June 2003

次世代ネットワークサービスモデルの専門展

IP.net JAPAN 2003

北村俊之

「加速するブロードバンド時代，次のステージがここから始まる!」をキーワードに，「IP.net Japan 2003」が2月26日(水)～28日(金)の3日間，東京ビッグサイトで開催された．主催は(株)リックテレコム．今年で第4回目を迎える同展示会では，最新のIPネットワーク技術やコンテンツの配信モデル，VoIPソリューションなどに関するデモンストレーションおよびセミナーが開催された．出展社数は72社，最終的な来場者数は10,097人となっている．

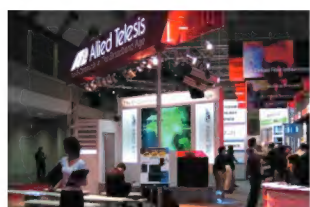
● IP.net Japan 2003

ソリトンシステムズで注目を集めていたのは，ATRICA社のメトロオプティカルEthernet A-8x00シリーズ(**写真1**)である．同製品は，バックボーンをEthernet化し，コスト削減を可能にする．同製品を利用したソリューションとして，通信インフラからユーザー課金までのトータルなビデオ配信システムの紹介を行っていた．

〔写真1〕**ATRICA社のEthernetスイッチA-8800**

アライドテレシスでは「The E-Community of The Broadband Age」をキーワードに，プロバイダや地方自治体向けのネットワークインフラ構築のためのソリューションを展示しており，レイヤ3ギガビットEthernetスイッチ，レイヤ3FastEthernetスイッチなど最新の製品展示とプレゼンテーションを行っていた．今後，付加価値の高いネットワーク環境が求めらる地方自治体などの関心が高いとのことであった(**写真2**)．

〔写真2〕**アライドテレシスのブース**

アプリケーショントラフィック管理製品では定評のあるF5ネットワークスジャパンでは，今回も同社製品の中核である「BIG-IPシリーズ」で来場者の注目を集めていた．また，同社のエンジニアが来場者の質疑に答える，ネットワーククリニックのコーナーも盛況であった．寺田電機製作所では，局内，データセンタ用の各種電力監視システム，インバータの展示を行っていた．コロケーションハウジングサービスなど自社以外の設備を預かるケースが増大するのにともない，電力の使用量把握と予防保全の観点から電源システムの常時監視への要求が高まっているとのことだった．また，高密度光終端架，メディアコンバータ，光スイッチングハブなどもあわせて展示されていた．

ピーエスアイでは，トランジションネットワークス社製のメディアコンバータを中心に展示を行っており，10/100/1000Mbpsの各種Ethernet，adm oc-3/oc-12，トークンリングなど多彩な製品群を用意しているのが特徴である．また，光装置/UTP変換のメディアコンバータも注目の製品であるという(**写真3**)．日本テレガードナーでは，レイヤ2光スイッチをはじめとして，ギガビットメディアコンバータ，FastEthernetメディアコンバータなどの光ファイバネットワークソリューションの紹介を行っていた．

〔写真3〕**ピーエスアイのメディアコンバータ**

スカイ・シンク・システムでは，LAN対応カメラを使用したIPネットワークベースの多地点映像システムの出展を行っていた(**写真4**)．net.com Japanは，SIPをベースとしたソリューションの展示およびデモを行っていた．また，ベルネット社の協賛をうけたVoIPのアウトソースサービスであるC-ASPも，来場者の注目を集めていた．

〔写真4〕**スカイ・シンク・システムの多地点映像システム**

内田洋行ではブロードバンド時代のインフラを有効活用した「ウチダブロードバンド映像配信システム」(**写真5**)の紹介を行っていた．こちらはADSL8/12Mを用いたサーバシステムとなっており，TVモニタとビデオプレーヤのリモコンでプログレッシブダウンロード再生を実現しているのが大きな特徴であるという．IPテレフォニーゾーンのネットワンシステムズでは，SkyWaveの各種製品を紹介していた．同製品群を利用すると中規模から大規模ネットワーク，キャリア，自治体まですべての環境において，SIPベースのVoIP環境をローコストで構築可能になるという．PDA用IPとSIPプロキシサーバ(**写真6**)を利用したデモを行っており，来場者の関心を集めていた．

〔写真5〕**内田洋行のブロードバンド映像配信システム**

〔写真6〕**SkyWaveのSIPプロキシサーバSkyProxy**

東洋紡エンジニアリングでは，ブロードバンドの設計，管理用CADシステムの展示を行っていた．同システムでは柱入力，管理やケーブル入力，管理などの施設管理から顧客管理まで，統合的な設計と管理ができるのが特徴．また，Webクライアントでは，PDAや携帯電話を利用して外部から地図情報やデータベース情報の検索，確認も可能となっている．

〔写真7〕**トーメンサイバービジネス/田村電機製作所のVDSL機器**

トーメンサイバービジネス/田村電機製作所(**写真7**)では，下り25Mbpsを実現させるだけではなく，最小サイズでの提供を実現している2Band VDSL機器に注目が集まっていた．また，業界最速クラスの4Band VDSL機器を初公開した．同製品では下り最大50Mbpsを実現しているとのことだった．日立電線では，長距離伝送装置GMXシリーズの新ラインナップ，メトロ用長距離伝送装置の発表および10Gビット対応/OC192対応装置の展示を行っていた．また，レイヤ2スイッチを利用したセキュリティ認証，リング構成の展示も行われていた．

Show report

Show & News Digest

日本テクトロニクス，モバイル分野向けプロトコルテスト計測事業に本格参入

- 日時：2003年3月12日（火）
- 場所：日本テクトロニクス本社（東京都品川区）

日本テクトロニクス（株）は，モバイル分野向けプロトコルテスト計測事業に本格参入することを発表し，新たにネットワーク計測営業部を設立した．

プロトコルテスト事業担当副社長のBob Agnesによると，すでに移動体通信サービスは公衆電気通信サービス収入の35％を占めており，これが2006年には45％まで増加すると予測されているとのことだ．このような背景から，同社はモバイルテスト分野を重視し，今回の発表となった．

同時に発表されたプロトコルテスタK1297は，従来のモニタ機能に加え，シミュレーション機能をサポートしている．また，QoSの測定などが可能なほか，複数インターフェースの同時テストが可能なことが特徴．

プロトコルテスト事業担当副社長のBob Agnes氏

T-Linux――MontaVista LinuxをT-Engine上へ移植

- 日時：2003年3月18日（火）
- 場所：帝国ホテル（東京都千代田区）

T-Engineフォーラムとモンタビスタソフトウエアによる共同記者発表会が開催され，組み込み向け開発プラットホームの“T-Engine”で動作するT-Kernel上へMontaVista Linuxを移植することで合意した．

T-LinuxはT-Kernel上のタスクの一つとして動作することにより，TRONのリアルタイム性能と，Linuxのネットワーク機能/豊富なアプリケーションを同時に使用できるようになるほか，eTRONセキュリティアーキテクチャによるハードウェアセキュリティ機能をLinux上からも利用可能になるなどのメリットをもっている．

今後の予定としては，2003年内にT-Linuxの仕様を決定し，2004年には製品を出荷するとのこと．

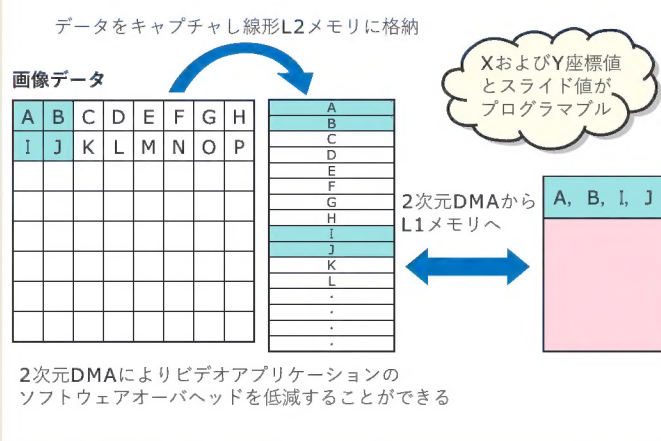
T-Engineフォーラム会長の坂村健氏（左）とモンタビスタソフトウエアジャパン（株）代表取締役社長の有馬仁志氏

アナログ・デバイセズ，Blackfin DSPファミリ新製品を発表

- 日時：2003年3月24日（月）
- 場所：帝国ホテル（東京都千代田区）

アナログ・デバイセズから，Blackfin DSPファミリの新製品「ADSP-BF533」，「ADSP-BF532」，「ADSP-BF531」の3種類が発表された．

同シリーズは最高600MHzで動作し（BF533），DPM（ダイナミックパワーマネジメント）機能により消費電力を280mWに抑えている．特徴的な機能としては，2次元DMAを搭載し，画像データの矩形転送を容易に行えるようにしている．対応OSとしてNucleus/ThreadX/Fusion RTOS/OSEKなどのほか，組み込みLinuxも予定しているとのこと．

価格は，600MHz品の$19.95～300MHz品で$4.95（10,000個時サンプル価格）．

2次元DMAの動作

ハッカーの常識的見聞録

30

広畑由紀夫

今月の常識

.NET Compact Framework がやってくる！

☆長くベータテストが続いていた.NET Compact Framework が，ついに正式版となって登場しました．では，開発環境などはどうなっているのでしょうか．

● 2003年4月24日

「.NET Compact Framework」は長らくベータテストが続きましたが，ようやく組み込み向けの.NET サポートが本格化してくるようです．4月24日から出荷開始されたのは英語版ですが，Visual Studio.NET は，以前にもまして強力にマルチランゲージをサポートしているので，日本語版も近く発売されることでしょう．

● 統合される開発環境　Visual Studio.NET 2003

すでにベータ版を試した人や，マイクロソフトの Web ページで情報を確認している人にはおなじみだと思いますが，現状の.NET Compact Framework は，編集ツールとして Visual Studio.NET 2003・Finalβ 版を使用しています．

.NET への対応を前提とした開発では，従来の Platform Builder も含めた Visual Studio.NET 2003 との協調開発環境により，ハードウェア開発からアプリケーション開発までの工程すべてにおいて TCO の削減などが図られることでしょう．やはり，組み込みにおいても，従来の企業向けの組み込み製品とは異なり，一般向けの Windows CE 端末やスマートフォンなどの製品開発においては，ユーザーが多様な使用法を求めるので，多彩なアプリケーション開発が要求されることになります．

● 組み込みでも多様なアプリケーションが必要

先日筆者は，PalmOS 5 の最新デバイスを一式10万円で購入しました．といっても筆者が使うためではなく，あくまで現場サポートのためだったのですが，担当者が必要だというので購入しました．

そこでいちばん問題になったのは，Palm デバイスに Web ブラウザとメールクライアントがインストールされていなかったことです．その他のツールについても，すべて PC を使用して CD-ROM からインストールするという仕様でした．

さて，購入後，担当者にそのまま渡してしまったのですが，その担当者が PC を使えない人で，「画面にメールがあるけど使えない」というので調べてみたら，メールのアイコンは準備されているものの，「CD-ROM からのインストールが必要」ということに気が付きました．そのほかにもアイコンが初めから表示されているものの，PC からインストールが必要なアプリケーションが多くありました．

一般向け携帯デバイスでインターネット対応を宣伝しているデバイスでは，メールクライアントや Web ブラウザなどは出荷時に組み込んでおいてほしいものだと感じました．

ユーザーの選択にまかせるというメーカーの立場などもあるとは思いますが，その結果として購入者が不便を被ることのないように気をつけてほしいと思います．

PalmOS 購入でのトラブルから，.NET Compact Framework と Platform Builder の組み合わせによって，ユーザーが安心して購入できる製品開発が加速してくれるのではないかと，非常に強い希望を感じました．

● 今後の希望

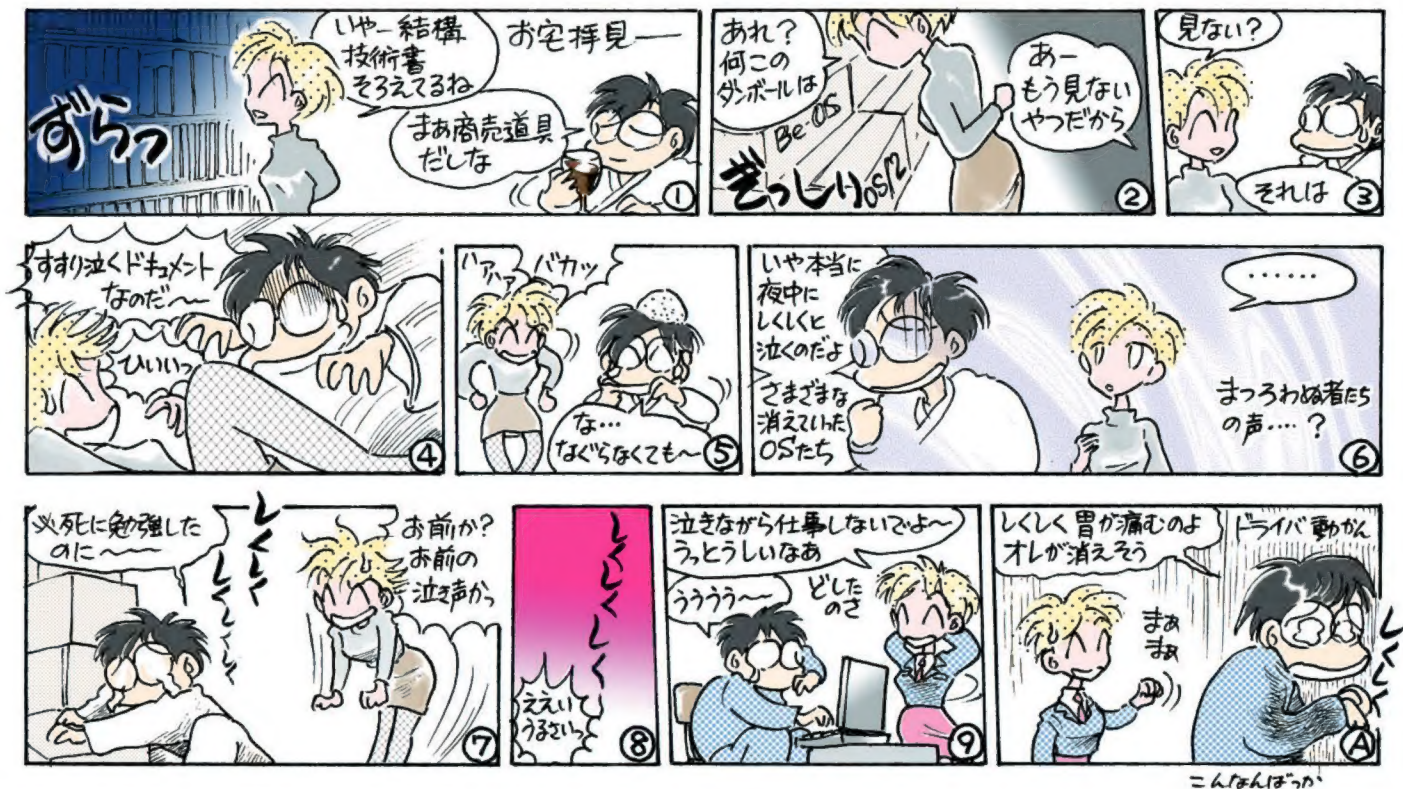
筆者としては，eMbedded Visual C++ が Visual Studio.NET にうまく統合され，Platform Builder との連携が Visual Studio.NET 単一で行われることを望んでいます．.NET を使用した開発で Windows とモバイルの連携がとりやすくなったとはいっても，今でも.NET とは離れた従来のコーディング方法が使われているからです．

旧来の C++ のコードなどは，Managed C++ でパックすることで再利用できるようになる可能性が非常に高いわけですが，やはり移行の手間がかかることに変わりはありません．たしかに，eMbedded Visual C++3.0 および 4.0 などで作成しているアプリケーションなどの移行に C# や VB.NET へ移し替えるのは容易ではないこともあるでしょう．

そうした面で，Windows アプリケーションの移植の容易性や互換性を高めるのみならず，今後はもっと低レベルの移植までのサポートに期待したいと思います．

参考 URL

1) Compact .NET Framework
http://www.microsoft.com/presspass/press/2003/mar03/03-19NETcfPR.asp
2) Visual Studio.NET 技術情報サイト
http://msdn.microsoft.com/vstudio/productinfo/vstudio03/overview/whatsnew.asp
http://msdn.microsoft.com/vstudio/device/compact.asp

ひろはた・ゆきお　OpenLab.

フジワラヒロタツの現場検証(70)

OSぼやき放談

技術者にとって，おさえておかねばならない新しい技術というものは数年に1回は出てくるのではないでしょうか．

古くはマイクロプロセッサが出てきたとき，筆者など，こんなに面白くて簡単なのに(!?)どうしてみんな慌てて，マイコンを理解できないと取り残されるなどと騒ぐのだろうと思っていました．――しかし，今になってわかります．若い頃は，歳をとってからの苦労などわからないものですね．

さて，ソフトウェア技術者にとってはOSの移り変わりというものも，いつになっても気になるものです．

OSは，その時々の流行というものがあり，そのたびに，不勉強な身にとっては落ち着かない気持ちにさせられます．

OSによっては，勉強しても主流にならず，仕事に役立たずにそのまま廃れていくモノも多く，勉強する側としてはなかなかリスキーなものです(使われないOSなんてねぇ)．

実際に使うことになってから勉強すれば無駄にはならないのですが，そうなってからでは間に合わないことが多いですし，少なくとも最小限度の部分くらい――そのOSが目的に使えるかどうかなども，OS選定から始まるような仕事では必要になりますから――は知っておかなければならないでしょう(どうも，筆者の考えは，技術的貧乏性というものかもしれません．OSなんてわずかな違いだけで，みんな原理は同じだといって泰然とできる方もいらっしゃいます．しかし筆者はそのわずかの違いでずいぶん痛い目に遭っているので，少なくとも技術職とは名ばかりの管理職になるまではそういわないほうが無難そうですね)．

CP/Mなんてのから始まって，iRMX，OS/2，Windows 3.0，OS-9/68000，BeOS，NeXTStep，ITRON，UNIX(BSD)，MacOS……ああもう嫌になります．なにより嫌になるのは，新しいOSが必ず新しい概念の理解を要求することです．その新しい概念も，少しはわかりやすく説明してくれればいいのに，われわれ多忙なエンジニアを精神的に痛めつけ，知的に服従させるためにわざわざ難解な回りくどい説明を行っているような気がします．ドライバの階層図なんて，OSのすべてがわかってからならわかるのですが，これから理解するためにはさっぱり役に立ちませんよね．

OS設計というのは，いわば世界を設計するようなものですから，「権力志向の持ち主がデザインするからこんな文章になるんだ！」などと，自分の理解力を棚に上げて，マニュアルを読みながらぶつぶつと文句をいったりします．

さて，最近の流行といえば，なんといってもLinuxですね．組み込み分野でもLinuxが俄然話題で，また新しいOSかい，とぶつぶつつぶやきながら，本誌のバックナンバーや解説書を読んでいらっしゃる読者諸賢も多くいらっしゃるのではとご同情いたします．考えてみればOSというのはどれも難物で，OSの枠にわれわれの頭とコードを合わせるように要求します．そういう意味で，Linuxのコード公開というのはやはりありがたいことではあります．ただし，だからといってすぐにコードを眺めて問題を解決できるわけではありませんけれどね．

藤原弘達　(株)JFP　エンジニア，漫画家

特集

ネットワークの基礎から音声通話への応用まで

TCP/IPの現在とVoIP技術の全貌

インターネットのインフラとして使われているTCP/IPも，誕生してから20年が過ぎた．基本的な部分は20年前から変化していないが，その細部を見ると確実にバージョンアップが行われている．そこで本特集では，TCP/IPの基礎と，TCP/IPに関する近年のトピックスなどをまとめて解説する．

また，TCP/IPの応用例の一つとして，昨今のブロードバンド環境の普及にともない，インターネット電話が注目されている．その根幹技術であるVoIPは，音声のディジタル化とエンコード，ディレクトリサーバによる通信対象の探索，インターネットにおけるQoSの維持，NATなど，ネットワーク技術の集大成ともいえるものだ．

そこで本特集では，VoIP技術についてVoIPの基礎知識からシグナリングプロトコル，音声CODEC，ネットワークの構築にいたるまでを徹底解説する．また特集の最後には，これらの知識を活用してLinuxを使用したVoIP電話の実験についても解説する．

特集第1章

インターネットの基本となるプロトコルを理解する
TCP/IPの基礎と現状

村上健一郎

インターネットの成立から20年の歳月が流れた．すなわち，インターネットの根幹技術であるTCP/IPも，それだけの月日を生き延びたということである．技術の世界で20年というと，10数世代にも相当する．これだけ長い間TCP/IPが陳腐化することなく使われているのは，その初期設計の良さと実装のしやすさのたまものである．そして，それらは偶然そのような設計になったのではなく，その背後の"文化"が強く影響した必然なのである．

本章では，このTCP/IP技術について基礎から解説し，VoIP技術を読み解くための道標とするとともに，TCP/IPが受けた文化的影響などについて考察する．

（編集部）

1 はじめに

ITバブルが崩壊したにもかかわらず，インターネット業界は騒々しい日々が続いている．xDSLやFTTHなどのブロードバンドが急速に普及しており，新たなアプリケーションが注目されている．その中の一つが，インターネットを使用して音声通信を行うVoIP（Voice over IP）である．

本号の特集はそのVoIPであるが，それに関する記事の前に，VoIP技術をスムーズに理解するための基礎となるTCP/IPの技術について説明する．VoIPのプロトコルには複数のものがあるが，まず，それらの背景となっているプロトコルの文化について言及する．

そして，ITU-TとIETFというメジャーな標準化組織，そこにおける標準化過程，標準化の結果としての規格書の構成について説明する．また，現在のインターネットのアーキテクチャと，それを実現するTCP/IPプロトコル群について，それらが設計された理念を出発点として解説する．その後，個別のプロトコルをレイヤ別に詳細に説明していく．

2 プロトコルと文化

2.1 二つの文化

通信プロトコルとは，通信手順のことである．もともとプロトコルは，外交儀礼とか議定書，協定などの意味をもっていた．それが後に，通信の分野で通信手順を意味する言葉として使用されるようになった．しかしプロトコルは，人間の情報処理を研究する認知科学の分野でも使用される．ここでの意味は，人間が何かの課題を達成するために行ったふるまいのことで，その過程で考えたことを言葉で言ってもらい，その行為を分析することをプロトコル分析と呼ぶ．だから，通信プロトコルの勉強をしようと思って，中身を確かめずに「プロトコル分析」という本を注文すると，発話データから何を読むかなどという本がきてしまう．ご注意を．

さて，ここでは，世の中のメジャーなプロトコルの背景となっている文化を紹介することにしよう．これは，なぜVoIPのプロトコルがいくつもあるのかを理解するのに役立つ．また，なぜインターネットが現在のようなアーキテクチャになったのかも理解できるであろう．

プロトコルの重要性は，それが共通に使用できることである．異なるコンピュータやOSがネットワークで相互に接続できないことには，それらの利用価値が下がってしまう．そこで，プロトコルの開発を行っている人や会社，政府は，自分の設計開発したプロトコルを世界標準にしようとする．そこには，大きくわけて二つの文化，すなわち，国際電気通信連合のITU-T（International Telecommunication Union - Telecommunication Standardization Sector）文化と，IETF（Internet Engineering Task Force）文化とが存在する．その文化ごとに，設計されるプロトコルは異なる特徴をもつ．

2.2 ITU-T文化

ITU-T（http://www.itu.int/）は，通信分野のプロトコル標準化を行う国連の一組織である．以前は，国際電信電話諮問委員会CCITT（仏語Comité Consultatif International Télégraphique et Téléphoniqueの略）と呼ばれていた．このメンバは各国の通信の主官庁（日本では総務省）や主管庁から承認された事業者，学術団体などの団体である．また，各国にITU-Tに対応する組織があり，日本の場合には，（社）電信電話技術委員会TTC（The Telecommunication Technology Committee）がそれにあたる．これは民間標準化機関であるが，総務省によって国際標準のITU-T勧告（ITUでは，プロトコルを標準として承認すると，それを勧告＝recommendationという形で出す）に基づく，唯一の国内標準作成機関として認定されている．

ITU-Tにおける標準化の特徴は，face-to-faceでじっくりと議論をすることである．そして，その承認は郵便投票によって行ってきた．このため，承認には平均で9か月を要していたが，最近では，インターネットの電子メールを利用したりして，これを短縮する努力がなされている．ITU-Tに対応する日本の組織TTCでは，国内標準を決めるだけではなく，そのプロトコルを国際標準としてITU-Tに提案したり，逆にITU-Tで標準となっ

たプロトコルに対応した国内のプロトコルの規定を行う．

ITU-T文化では，プロトコルの標準化はきわめて真面目に計画的かつ紳士的に行われる．標準化に参加している人たちは，国の代表として，自国の事情や技術を勧告にできるだけ盛り込もうとして努力する．この結果，できあがった規格は豪華絢爛で機能に富み適用範囲が広いものとなる．しかし，そのために大規模で複雑化することもあり，全部を実装することが困難なこともある．

ITU-Tの勧告名は，分野別にシリーズ化されて分類されている．分野は勧告名の頭文字によって識別される．代表的なシリーズ名を**表1**に示す．たとえば，VoIPに使用されているプロトコルにH.323勧告というのがあるが，これは，オーディオビジュアル/マルチメディア関連の勧告ということがわかる．また，G.709であれば，伝送システムか音声符号化の勧告であるということがわかる．

わかりにくいのが，改訂されても勧告が同じ番号をもつことである．番号だけではいったいどの勧告の話をしているのかがわからない．そこで，勧告に年度や勧告書の色をつけて区別をすることがある．たとえば，1993年度に承認されたG.708規格ならばG.708(93)と記述することがある．また，1988年の規格ならばその勧告書の色が青なのでブルーブック版と呼ぶこともある．場合によっては，バージョン番号を規格名の後につけることもある．たとえば，H.323のバージョン2の場合は，H.323.2という具合である．

ITU-Tの標準化組織の構成を**図1**に示す．それぞれの勧告は，各研究委員会SG(Study Group)によって標準化されたものである．たとえば，H.323を担当しているのは，SG16である．SGの下には作業部会WP(Working Party)がある．

2.3 IETF文化

1）標準化組織

一方，IETF(Internet Engineering Task Force)のほうは，誰でも参加できる．手をあげれば，その時点からプロトコルの提案ができる．こちらのほうはITUのような公的標準(De Dure Standard)ではなく，業界の事実上の標準(De Facto Standard)を決める．IETFはインターネット協会ISOC(Internet Society)配下の組織である．ISOCのURLは，`http://ww.isoc.org/`である．この構成を**図2**に示す．標準化のおおまかな方向性は，

〔表1〕ITU-Tの勧告シリーズ

勧告シリーズ名	内　容
Hシリーズ	オーディオビジュアル/マルチメディア関連
Gシリーズ	音声符号化方式関連
Qシリーズ	ISDNと電話網の交換方式および信号方式に関する勧告
Vシリーズ	アナログモデム関連
Tシリーズ	G3/G4ファクシミリ関連
Xシリーズ	データ通信網に関する勧告
Iシリーズ	サービス総合ディジタル網(ISDN)

IAB(Internet Activity Board)委員会が示す．IETFは，標準化を行う分野(エリア)別にいくつものワーキンググループWG(Working Group)に別れている．ワーキンググループのメンバに資格はない．ボランティアベースである．規格を提案したい個人は，インターネットドラフトと呼ばれる文書を提出する．これは，寄書(contribution)とも呼ばれる．

提案や議論はおもにメーリングリストで行われる．また，年に3回の顔をあわせたミーティングがある．なお，IESG(Internet Engineering Steering Group)は，IETFの各分野の代表で構成される委員会で，ドラフトを公式文書RFC(Request For Comments)として承認したり，IETF活動の管理を行う．RFCについては以下に詳しく説明する．

2）RFC

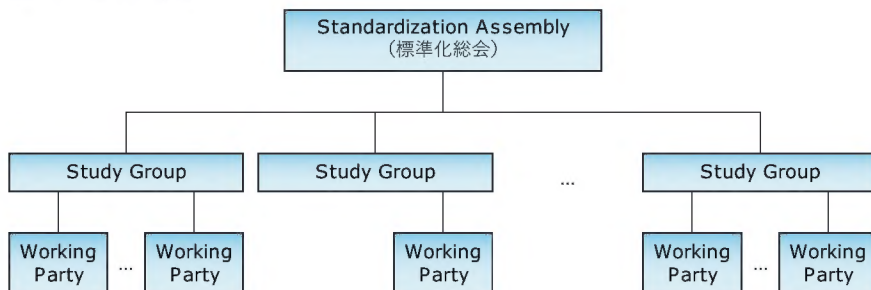
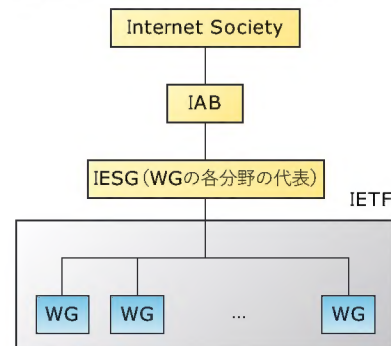
IETFの公式文書RFCは，もともと，“このプロトコルのドキュメントにコメントをください”という意味でRequest for Commentsという名前がつけられた．RFCを公開し，それに対して議論とコメントをつのったのである．ここでは，RFCの種類と標準化過程について説明する．

インターネットドラフトは，標準化のプロセスを経るとRFCという番号のつけられた文書となる．たとえば，RFC791はIPのプロトコルを規定するRFC，RFC793はTCPのプロトコルを規定するRFCである．ただし，すべてのRFCが標準というわけではなく，いろいろな種類のものがある．これらを**図3**に示す．たとえば，標準化の過程を経ずにプロトコルなどの情報を公開するInformational RFC，研究の結果や実験的な情報を公開

〔図1〕ITU-Tの構成

〔図2〕ISOCにおける標準化組織

Internet Society
IAB
IESG(WGの各分野の代表)
IETF
WG　WG　…　WG

New Products ― カード型無線機器向け薄さ1mmのセラミックフィルタ
(株)村田製作所は，薄さ1mm以下という10.7MHz帯セラミックフィルタ「SFECD10M7シリーズ」を発売した．3.45×3.1×1.0mmのSMDタイプで，セラミックスの材質を改良して強度を高めた薄型外装基板を採用することにより，同社従来品の2/3の厚さを実現した．

するExperimental RFC，プロトコルの適用方法や利用ガイドラインなどの「その時点での最良の方法」を示すBCP（Best Current Practice）のRFCなどがある．また，標準化を行うRFCの場合には，以下のステップを経る．

i）Proposed Standard

標準化するプロトコルの最初のステップとなるRFCである．複数の組織で独立した実装があり，相互接続性の確認がなされていることが条件となる．

ii）Draft Standard

この段階ではすでにプロトコルが広範囲で使用されている．Draft Standardは，仕様として十分に成熟したもので，実質的には標準とみなすことができる．

iii）Standard

名実ともに標準プロトコルであり，STD（STandarD）シリーズの番号が割り振られる．しかし，現在，STD番号のあるRFCは62個しかない．

すでに標準であったものが，別の仕様によって使われなくなった場合などには，そのRFCは，Histrical RFCとなる．これらの種別は各RFC文書のヘッダに書かれている．これらすべてのインターネットドラフトやRFCはIETFのWebページで見ることができる．URLはhttp://www.ietf.org/である．

なお，毎年4月1日発行のRFCはまじめに読んではいけない．たとえば，RFC3251は，IP上で電気を運ぶプロトコルを規定している．RFC3093はファイアウォールでTCP/IPを透過的に通すためのプロトコルを規定している．また，RFC1149は伝書鳩によるIPデータグラムの伝送について既定している．このような点からも，ITU-TとIETFの雰囲気の違いがわかるであろう．

ITU-Tのほうのドキュメントと異なり，RFCでは，プロトコルが更新されると新たな番号を付与する．新たなRFC文書のヘッダには，どのRFCを置き換えるのかを明記する．したがって，同じプロトコルに異なるバージョンがあっても混同することは少ない．**表2**に代表的なプロトコルの例を示す．なお，各プロトコルは一つだけのRFCで規定されているわけではなく，複数のRFCにわたっていることがある．ここでは，おもなものだけを示している．

〔図3〕RFCの種別と標準化過程

3）ラフコンセンサスとランニングコード

IETFでの標準化のポリシは，二つのキーワードで表すことができる．それは，ラフコンセンサス（Rough Consensus）とランニングコード（Running Code）である．前者は，キーになる人たちの間でだいたい合意できたようであれば，それを承認しようということである．実際，バブル前は，このようにおおらかなものであったが，バブル以降は，騒々しくなってようすがかわってきている．すなわち，参加者が著しく増え，個人よりも企業の戦略が強く働くようになって，いつまでたってもコンセンサスが得られないという混沌とした状態になっているグループもある．

後者のランニングコードのほうは，インターネットの標準化では実際に動くコード（プログラム）が最重要であるということを意味する．しかも，独立した実装が複数なければ，標準化の舞台にのせてはもらえない．つまり，標準化が開始されたならば，その時点ですでに相互接続のできる実装があり，ドキュメントには相互接続を妨げるようなあいまいさがないということが検証されている．これらは，ITU-Tとは大きく異なる点である．この比較を**表3**に示す．

3 インターネットの設計理念とアーキテクチャ

TCP/IPのようなプロトコル群を設計する場合には，それが実現するシステムの果たすべき機能を明確にし，それをどのような形でまとめるかという設計理念がまず必要となる．逆に，そ

〔表2〕代表的なプロトコルのRFC番号

プロトコル	おもなRFC	旧版RFC
IP（Internet Protocol）	RFC791	RFC760
ICMP（Internet Control Message Protocol）	RFC792	RFC777，760
TCP（Transmission Control Protocol）	RFC793	RFC761
UDP(User Datagram Protocol)	RFC768	−
DNS（Domain Name System）	RFC1035	RFC882，883，973
HTTP（Hyper Text Transfer Protocol）	RFC2616	RFC2068
SMTP（Simple Mail Transfer Protocol）	RFC2821	RFC821，974，1869
TELNET（TELecommunication NETwork）	RFC854	−
FTP（File Transfer Protocol）	RFC959	RFC765
SIP(Session Initiation Protocol)	RFC3261	RFC2543

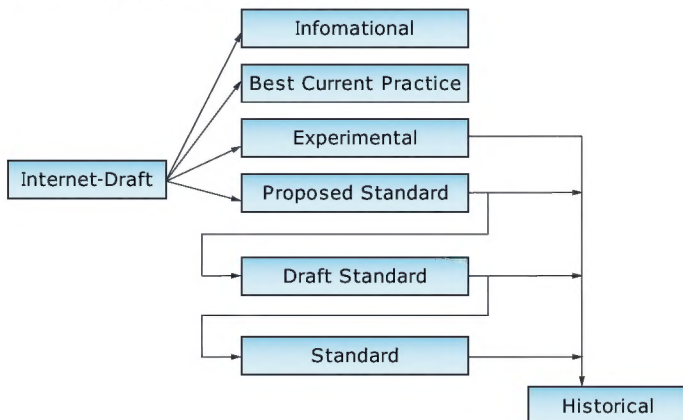
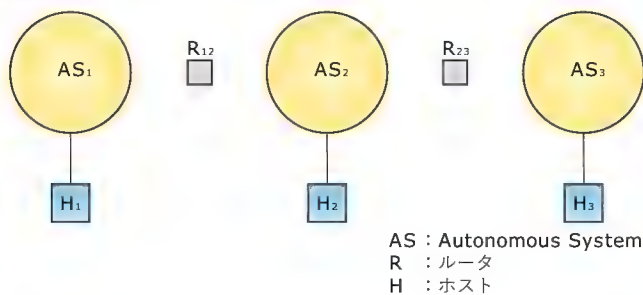
〔表3〕ITU-T文化とIETF文化

	ITU-T文化	IETF文化
優先事項	プロトコルの仕様	実装（ランニングコード）
プロトコル承認方法	投票	ラフコンセンサス
ドキュメントの入手	有料	無料
標準の種類	De Dure Standard（国際標準）	De Facto Standard（事実上の標準）

New Products ―国産の商用SystemCデバッガソフトウェア
キャッツ(株)は，業界初となる国産の商用SystemCデバッガソフトウェア「XModelink（クロスモデリンク）SystemC Debugger」を5月1日に発売する．SystemCデバッグとVisualC++.netの双方への対応機能，ハード/ソフト混在型のシステム全体設計/検証機能などが特徴．

〔図4〕インターネットのモデル

〔図5〕TCP/IPプロトコル群のレイヤ構造

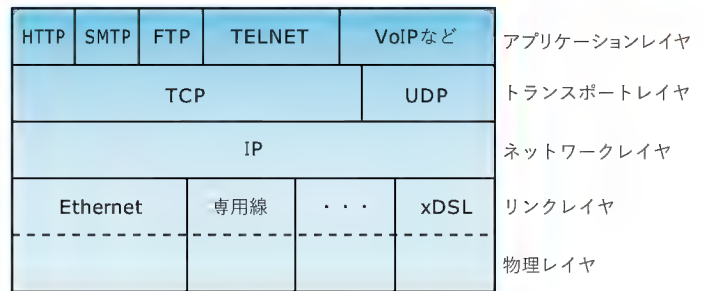
HTTP	SMTP	FTP	TELNET	VoIPなど	アプリケーションレイヤ
TCP				UDP	トランスポートレイヤ
IP					ネットワークレイヤ
Ethernet	専用線	・・・	xDSL		リンクレイヤ
					物理レイヤ

の理念の理解なしには，インターネットやTCP/IPのプロトコル群がなぜ今の形となっているのかを理解することは難しい．

たとえば，“インターネットはパケットがなくなって信頼性が低いから問題だ”という意見を耳にすることがある．しかし，そう思うのは早計である．なぜならば，帯域を保証し機能満載で複雑なネットワークの対極として，帯域を保証せずに機能を簡単にするという理念の下にインターネットを設計したからである．すなわち，ここでの理念は，ネットワークの中核部分は簡単にすべきというものである．そして，複雑な機能のほうはネットワークに接続されるホスト側にまかせてしまう．その結果，インターネットはひたすらパケットを転送することに専念し，エラーの訂正も，フロー制御も行わないものとなった．輻輳(ふくそう)（ネットワークの過負荷）が発生したら，単にパケットを捨ててしまう．

また，電話のように通信に先だって相手までの通信路を設定するコネクション指向のネットワークではなく，パケットを単にネットワークに渡せば，指定された相手のコンピュータまでただちにパケットを転送してくれるというコネクションレス型のネットワークとした．

インターネットにはもう一つの理念がある．それは，インターネットの名前の由来となった「ネットワークのネットワーク」という考え方である．つまり，inter-networkである．

インターネットは1983年にARPA（Advanced Research Project Agency）[注1] Internetとして始めて世に姿を表わしたが，その前身のARPAnetでは，NCP（Network Control Protocol）というプロトコル体系を採用していた．このプロトコル体系では，ネットワーク全体を一つのシステムとして管理する必要があった．しかし，このようなネットワークでは，規模の拡大にともなって管理も複雑となり，破綻する恐れがある．そこで，規模が拡大しても管理がうまくいくように，各組織が管理するネットワークを相互接続した集合体とした．これがインターネットアーキテクチャである．各組織が管理するネットワークのことを自律システムAS（Autonomous System）と呼ぶ．

インターネットは，この自律システムをパケット交換機であるルータ（Router）で相互接続した集合体である．**図4**にこのインターネットのモデルを示す．自律システムとは，同一の組織によって管理運営されるネットワークのことである．

たとえば，各大学や企業のネットワークは自律システムである．ISP（Internet Service Provider）も自律システムである．ルータは，自律システム間でパケットを中継する．自律システムの内部は，Ethernetのようなネットワークセグメント，それに接続されたサーバやPCなどのホストから構成される．しかし，自律システムの内部が，さらにルータで接続された自律システムの集合になっており，inter-networkを構成している場合もある．たとえば，大学のキャンパスネットワークは，その内部が各学部の自律システムから構成されており，inter-networkを形成していることが多い．このように，インターネットは入れ子構造をもつ．

4 TCP/IPプロトコル群のアーキテクチャ

4.1 プロトコルのレイヤ構造

インターネットを実現すべくTCP/IP（Transmission Control Protocol/Internet Protocol）プロトコル群が設計された．「プロトコル群」というくらいだから，プロトコル単体のものではなく，一連のプロトコルの集合体である．各プロトコルが連携することによって全体としてインターネットの通信機能を実現する．

インターネットのプロトコル群は，ちょうどソフトウェアが機能別にモジュール化されて実装されるのと同じく，レイヤ（層）別にモジュール化されている．これを**図5**に示す．このプロトコル群は，五つのレイヤから構成される．それぞれのレイヤには，複数のプロトコルが属する場合がある．これらは，使用目的によって使い分けられる．ここでは各レイヤのプロトコルのうち代表的なものだけを示している．

下位のレイヤは，上位のレイヤにサービスを提供する．上位のレイヤはそのサービスを使用して，さらに上位のレイヤにサービスを提供する．

たとえば，リンクレイヤは，同一Ethernetセグメント上の隣接するホスト間のパケット転送サービスをネットワークレイヤに提供し，ネットワークレイヤは，これを使用して，異なるEthernetセグメント上にあるインターネットのホスト間（エンドツーエンド）のパケット転送サービスをトランスポートレイヤに

注1：高等研究計画局ARPAは国防高等研究計画局DARPA（Defence Advanced Research Project Agency）といった時期もあり，米国国防総省の機関である．国防に関連した研究プロジェクトを計画したり，それを管理する．

New Products ― ±1℃の精度とPWM出力を備える小型ディジタル温度センサ
アナログ・デバイセズ社は，工業用プロセス制御，計装機器，医用機器向け電池駆動ポータブルアプリケーション向けの低消費電力，小型パッケージのディジタル温度センサ「TMP05」「TMP06」を発売した．1線パルス幅変調（PWM）出力を備え，精度は±1℃（0～70℃にて）．価格は$0.95（1,000個時）．

提供する．なお，インターネットでは，パケットをレイヤごとに区別するため，リンクレイヤにおいてはフレーム(frame)，ネットワークレイヤにおいてはデータグラム(datagram)，トランスポートレイヤにおいてはセグメント(segment)と呼ぶ．

4.2 物理レイヤとリンクレイヤの概要

物理レイヤは変調方式や光ファイバなどの伝送媒体などの規定をする．また，リンクレイヤは，同一ネットワークセグメント上のホスト間でのフレームの転送を規定する．リンクレイヤの代表的なものとして，Ethernet，専用線，モデムを使用した電話網，ISDNなどのリンクがある．インターネットでは既存のリンクをそのまま使用するので，RFCでは，それらの使い方は規定しても，物理レイヤやリンクレイヤのプロトコルは規定しない．このため**図5**では，物理レイヤとリンクレイヤとを一体化して示している．

リンクレイヤの機能は，同一回線上の隣接するルータやホストまでフレームを転送することである．ここで重要なことは，インターネットがどのようなリンクでも使用できることである．すでに述べた4月1日付けのRFC1149“伝書鳩によるIPデータグラムの伝送”を実際にやった人もいるくらいである．このようにさまざまなリンクが使用できるのは，IPデータグラムをリンク固有のフレーム内のデータとして転送するからである．これをエンキャプシュレーションと呼ぶ．このようにするのは，次の理由による．

もしも，インターネットプロトコルとして物理レイヤ/リンクレイヤのプロトコルを規定すれば，プロトコル群はきわめて大きなものとなる．また，すでに存在する規格を使用するとしても，それぞれのリンクの規格と渾然一体となったプロトコルを規定していたのでは，プロトコル群の複雑化が進む一方である．

そこで，どのようなリンクのプロトコルでも使用できるように，TCP/IPのパケットをリンクのパケットにどう入れるかというエンキャプシュレーションの方法(5節を参照)とTCP/IPのアドレスとリンクのアドレスとの対応付けを動的に行うアドレス解決の方法(7.4節を参照)だけを規定することにした．アドレス解決のおかげで，リンクのアドレスの構造とインターネットのアドレスの構造とをまったく独立したものにできる．したがって，異なるアドレス構造の新たなリンクの規格が出現してもすぐに対応できる．

4.3 ネットワークレイヤの概要

ネットワークレイヤは，インターネットレイヤと呼ばれることもある．ネットワークレイヤは，トランスポートに対してインターネットのホスト間(エンドツーエンド＝end to end)のデータグラムによる転送サービスを提供する．そのために，下位のリンクレイヤのサービスを使用する．このレイヤには，IP(Internet Protocol)のプロトコルがある．IPのプロトコルでは，IPデータグラムの喪失などの誤りが発生しても，回復を行わない．

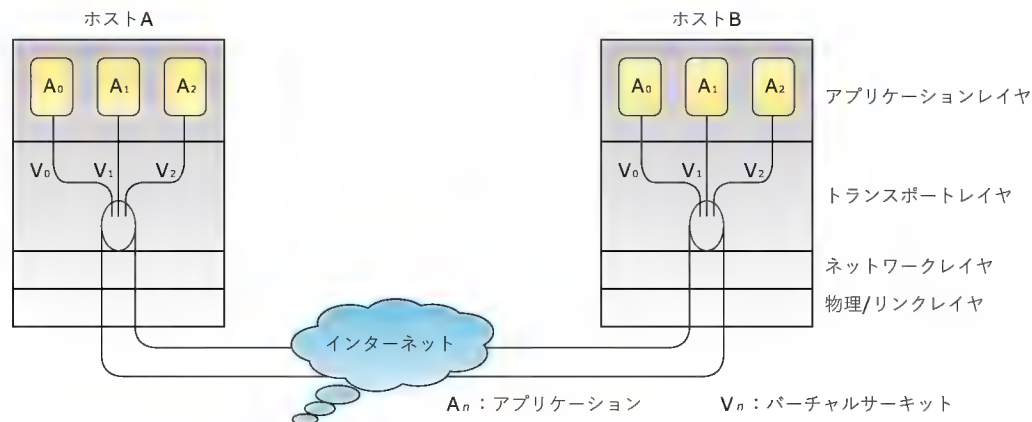
個々のホストやルータは，回線を接続するインターフェースごとに，インターネット内で一意に識別できるIPアドレスをもっている．転送元のホストは，データをIPデータグラムに分割して転送する．その際に，転送先のホストに割り当てられたIPアドレスを，転送先アドレスとしてIPデータグラムに埋め込む．そして，回線に送り出して隣接するルータに渡す．ルータには複数のリンクが接続されている．あるリンクからフレームを受信すると，その中のIPデータグラムを取り出し，その転送先IPアドレスを元に，次に渡すべき隣接したホストあるいはルータを決定する．そして，そこへ接続されているリンク固有のフレームにIPデータグラムをエンキャプシュレーションして転送する．

このようにしていくつものルータによって中継されたIPデータグラムは，最終的に転送先のホストに到着する．転送された一連のIPデータグラムは，転送先のホストによって元のデータに復元される．たとえば，**図4**のホストH_1で，IPデータグラムに分割されたデータは，ネットワーク上を二つのルータR_{12}，R_{23}によって中継されながら転送先のホストH_3へ転送されていき，そこで元のデータに組み立てられる．

4.4 トランスポートレイヤの概要

このレイヤでは，通信を行う二つのホスト内のアプリケーションプロセス間に，全二重の仮想的な専用の通信路(バーチャルサーキット)を提供する．これを**図6**に示す．双方のホスト間でのパケットの転送には，下位のネットワークレイヤのサービスを使

〔図6〕バーチャルサーキット

New Products ― DSLモデム，白色LED向け昇圧型DC-DCコンバータ
ナショナルセミコンダクタージャパン(株)は，小型パッケージを採用し高電力密度を実現した昇圧型DC-DCコンバータ「LM2731」「LM2733」を発売した．カレントモードコントロール，高スイッチング周波数を特徴としており，価格は¥131(1,000個時)．

用する．このレイヤには，TCP（Transmission Control Protocol）やUDP（User Datagram Protocol）のプロトコルがある．

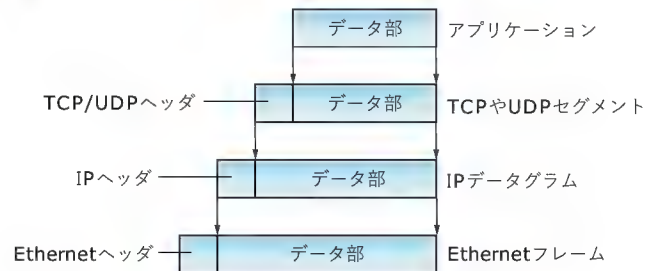
TCPでは，ネットワークレイヤにおけるパケットの欠落や損傷などの転送中のエラーを検出し，再転送による誤りからの回復を行う．このため，TCPはデータのバイトごとに番号を付けて転送先のホストへ送る．そのホストでは，受信した最後の番号に1を加えた番号（すなわち，次に受信を期待するデータの番号）を送信側のホストへ応答として返す．転送元のホストでは，一定時間内に応答が返ってこない場合には，パケットが失われたものと判断して再転送する．このエラーからの回復に加え，インターネット内部や相手のホストの負荷を推測し，ネットワークに過負荷を与えず，かつ，最大の転送速度が得られるようにパケットの転送を調節する．

なお，UDPのプロトコルはTCPと異なり，応答確認を行わない．このために，ホスト側の処理が軽い．そこで，連続してデータが転送され，しかも，誤りからの回復の必要のないVoIPの音声や動画などのストリーム転送に使用される．

4.5　アプリケーションレイヤの概要

アプリケーションレイヤは，下位のトランスポートレイヤのサービスを利用し，ユーザーに対して特定のアプリケーションのサービスを提供する．このレイヤには，Webを閲覧するためのHTTP（Hyper Text Transfer Protocol），リモートログインを行うための仮想端末プロトコルTELNET（TELecommunication NETwork），ファイル転送を実現するFTP（File Transfer Protocol），メール転送を行うSMTP（Simple Mail Transfer Protocol）などのプロトコルがある．これらは，トランスポートレイヤのTCPプロトコルが提供するバーチャルサーキットを使用する．

一方，UDPを利用するものとして，インターネット電話VoIPの音声の伝送がある．しかし，複数あるVoIPのプロトコルの中には，電話をかけたり切ったりする処理（呼制御）に関して，TCPのプロトコルを使用するものがある．

5　エンキャプシュレーションとフラグメンテーション

パケットは，データを転送する場合の単位である．これは，転送先や転送元のホストを示すアドレスなどを含むヘッダ領域と，データを入れるデータ領域とから構成される．データ領域は，ペイロードと呼ばれることもある．上位レイヤのパケットは，下位レイヤのパケットのデータ領域に入れられて転送される．このエンキャプシュレーションの関係を**図7**に示す．転送元のホストのアプリケーションはデータを分割し，TCPのセグメントに入れる．セグメントはIPのデータグラムに入れられる．データグラムはリンクレイヤのフレームに入れられる．そして，このフレームが回線上を転送される．

〔図7〕エンキャプシュレーションの例

フレームを受信したルータは，データグラムを取り出し，そのヘッダにある転送先アドレスから次に渡すべきルータを決定し，そのルータへ接続されている回線に合ったフレームにエンキャプシュレーションして送り出す．これによってデータグラムがエンドツーエンドで中継される．このようすを**図8**に示す．

転送の際，ホストやルータは，データグラムを送り出す回線のリンクプロトコルが許しているフレーム内の最大データ長を考慮する．その最大データ長をMTU（Maximum Transmission Unit）と呼ぶ．IPデータグラム長がMTUを越える場合には，それを複数のデータグラムに分割して，それぞれがMTU内に収まるようし，複数のフレームにエンキャプシュレーションして転送する．これをフラグメンテーションと呼ぶ．これによって生じたフラグメントは，転送先のホストで元のデータグラムに組み立てられる．

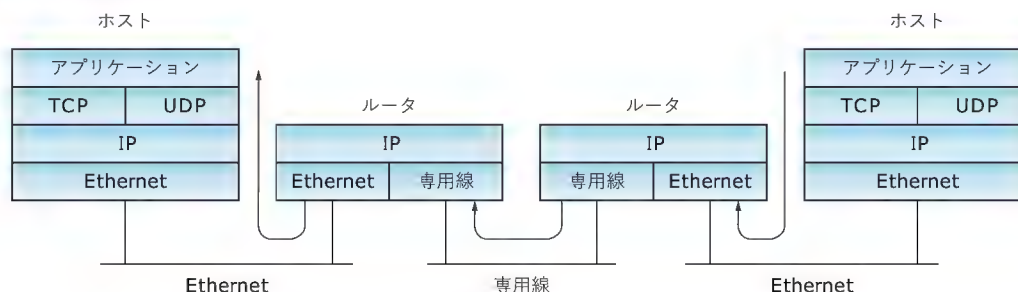
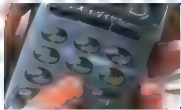
6　リンクレイヤのプロトコル

6.1　Ethernet

1）フレーム形式

リンクレイヤにはEthernet，専用線，xDSLをはじめ，さま

〔図8〕エンドツーエンドの転送

New Products ―ハード性能，拡張性が向上した手の平サイズのマイクロサーバ「OpenBlockS266」
ぷらっとホーム（株）は，小型で廉価なマイクロサーバ「OpenBlockS（オープンブロックス）シリーズ」の最新機種「OpenBlockS266」を発売する．CPUを266MHzに高速化したほか，従来からの手の平サイズ，ファンレス，2ノードのLAN，DHCPサーバ機能，簡易ファイアウォール機能を継承している．価格は￥39,800．

ざまなプロトコルがある．一般に，リンクレイヤのプロトコルが使用するフレームは，ヘッダ部とデータ部とから構成される．データ部にはIPデータグラムがエンキャプシュレーションされる．データ部の最大長MTUはリンクによって異なる．たとえば，Ethernetの場合には1500バイトである．このEthernetのフレームを**図9**に示す．

ヘッダ部は，Ethernetの場合には，転送先アドレス，転送元アドレス，プロトコルなどから構成される．また，フレームの最後には，エラーを検出するためのFCS(Frame Check Sequence)がつけられている．送信の際に，FCSを除くフレーム全体を対象としてCRC(Cyclic Redundancy Check)の生成多項式による計算をし，結果をここに入れる．受信側でも同じ計算をして比較し，一致しない場合にはエラーとみなす．

プロトコルのフィールドには，データ部に何の上位レイヤのプロトコルのパケットが入っているかを示す識別子が入っている．たとえばIPv4の場合には，0800(16進)が入れられる．なおヘッダの構成には，ここで説明しているオリジナルのEthernetと，それを基に米国電気電子技術者協会IEEE(Institute of Electronical and Electronics Engineers)が標準化したIEEE802.3標準の形式とがある．後者の形式では，プロトコル識別子ではなく，データ長がここに入れられる．詳しくは，参考文献1)の第3章を参照していただきたい．

2）アドレス

アドレスは，MAC(Media Access Control)アドレスとも呼ばれる．これは全長が48ビットで，上位24ビットが装置を製造した会社を識別する番号，下位24ビットは，その会社内で製造した装置ごとの識別番号になっている．前者は，IEEEによって管理され，各会社に割り当てられている．後者は，どの装置も一意のアドレスをもつように会社内で管理される．このため，各装置は世界で唯一のアドレスをもち，アドレスの衝突が発生しない．製造者番号に関しては，`http://standards.ieee.org/regauth/oui/index.shtml`のWebページをアクセスすれば，検索したり一覧表を閲覧できる．たとえば00-50-da-b7-ee-2dというMACアドレスは，先頭24ビットである00-50-daから，製造した会社が3COM社ということがわかる．

なお，アドレスの第一バイト目の最下位ビットは，ブロードキャストやマルチキャストと通常のユニキャストの区別を示すビットである．これが，0の場合には，このアドレスをもつ唯一のホストで受信される．また，これが1の場合には，特定のグループに属する複数のホストが受信するマルチキャストである．それ以外のビットもすべて1の場合には，すべてのホストで受信されるブロードキャストを示す．

3）送信方式

転送元のホストは，転送先のホストのEthernetアドレスをヘッダに入れてEthernet上に送信する．Ethernetの場合には，多重アクセスが可能であり，一つのEthernetセグメントに複数のホストを接続することができる．

たとえば，通常，多数のホストがツイストペアケーブルでEthernetハブに収容されている．送信側では，他が送信していないのを見計らって送信するが，同時に他のホストも送信したために衝突が発生することがある．このため，送信中は信号を監視し，衝突を検出した場合には送信を停止する．そして，時間を見計らって再送信する．これは，送信が成功するまで繰り返される．この方式をCSMA/CD(Carrier Sense Multiple Access/Collision Detection)，つまり，キャリア検知多重アクセス/衝突検出方式と呼ぶ．

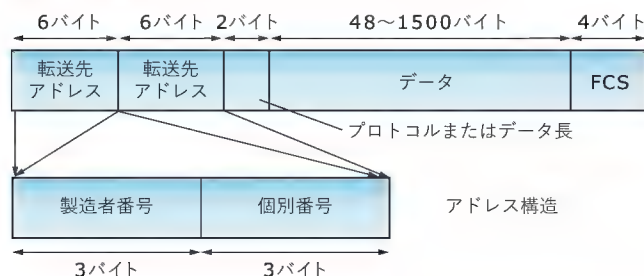
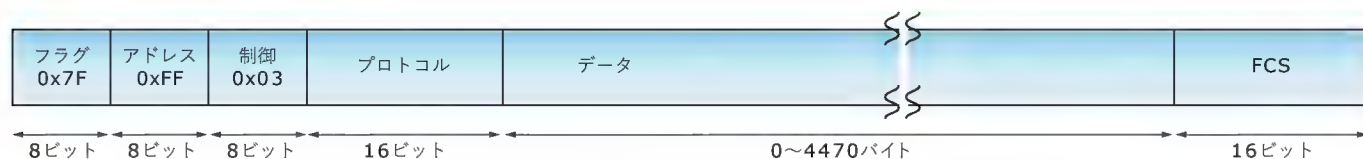
6.2 専用線

専用線の場合にはEthernetと異なり，接続はポイントツーポイントである．専用線上でデータを送るためには，HDLC(High-level Data Link Control)フレームもしくはそれを基にしたフレームがよく使用される．これも，ヘッダ部とデータ部から構成されている．たとえば，RFC1661，1662などで規定されているPPP(Point-to-Point Protocol)[2]もHDLCフレームをベースとしており，そのフレーム形式は，**図10**のようになっている．転送先アドレスは常にブロードキャスト(すべてのノードが受信するアドレス)を示すオール1である．Ethernetと異なり，データ部に上位レイヤのデータグラムを入れて送信するだけで相手のホストで受信される．MTUは利用する回線の種類によって異なる．たとえば，155Mbps以上の超高速専用線の場合，4470バイトである．

なお，最近のブロードバンドで使用されているxDSLやCATVなどのリンクレイヤのプロトコルに関しては，誌面の都合上，省

〔図9〕Ethernetのフレーム形式とアドレスの構造

〔図10〕PPPのフレーム構造

Information ── 松下電器がTOPPERSプロジェクトの開発成果を採用
ITRON仕様OSなどを開発しているTOPPERS(トッパーズ)プロジェクトの開発成果である「TOPPERS/JSPカーネル」が，松下電器のカラオケマイク「DO!KARAOKE」に採用された．TOPPERS/JSPカーネルは，μITRON4.0仕様のスタンダードプロファイル規定に準拠したリアルタイムOSである．

略する．これらについては，本誌2001年9月号特集記事などを参照していただきたい．

7 ネットワークレイヤのプロトコル

7.1 データグラムの形式

ネットワークレイヤにはIPv4（IPバージョン4）やIPv6（IPバージョン6）のプロトコルがある．ここでは，現在のインターネットで使用されているIPv4について説明する．**図11**にIPデータグラムの形式を示す．これは左側から転送される．データグラムは，ヘッダ部とデータ部とから構成される．データ部には，上位レイヤのTCPあるいはUDPセグメントがエンキャプシュレーションされる．

ヘッダ部を詳細に示したのが**図12**である．ここでは，パケットの先頭を左上に書いており，転送は上から下へ行われる．また，32ビット幅でフィールドを示している．以下では，おもなフィールドについて説明する．まず，バージョン（version）フィールドには，IPv4の場合，4の値（0100）が入る．また，ヘッダ長IHL（Internet Header Length）には，4バイトを一単位としたIPヘッダの長さが入る．オプションがない場合には，この長さは20バイトになるので5という値（0101）が入る．パケット長（Total Length）フィールドは，このデータグラムの長さ（単位はバイト）である．その最大値は，65,535である．

識別子（Identification）はパケットごとの識別に使用され，それぞれのパケットは異なる値をもつ．なお，フラグメンテーションによってIPデータグラムを複数のフレームで運ぶ場合，すべてのフラグメント化されたデータグラムの識別子は元のデータグラムのそれと同じ値をもつ．転送先のホストで元のデータグラムに復元するが，この識別子を参照すれば，どれが元のデータグラムの一部なのかがわかる．フラグメントオフセット（Fragment offset）は，このデータグラムが，フラグメントされる前のデータグラムのどの位置にあったものかを先頭からのバイト位置で示す．

生存時間TTL（Time to Live）は，ルータによって中継されるたびに減らされる．そして，0になった時点で，当該データグラムは破棄される．これは，ループが発生しても永遠に中継されることがないようにするためである．プロトコル（Protocol）フィールドは，データ部に入っているセグメントがどの上位プロトコルのものであるかを示す．たとえば，TCPの場合6（バイナリで0000 0110という値が入り，UDPの場合には，17（バイナリで0001 0001）という値になる．

〔図11〕IPデータグラム形式

〔図13〕IPアドレスの構造

ヘッダチェックサム（Header Checksum）は，転送誤りを検出するためのものである．これは，ルータによって中継されるたびに再計算される．それは，TTLフィールドが変更されるためである．チェックサムの計算は次のようにして行う．まずチェックサムのフィールドを0にして，IPヘッダ部のみを対象として16ビット単位で1の補数の和を求める．さらに，その値の1の補数を計算してチェックサムフィールドに入れる．チェックサムの計算方法は，参考文献3）の付録4が詳しい．

転送元アドレス（Source Address）は転送元のホストのIPアドレスを，転送先アドレス（Destination Adderss）は転送先のホストのIPアドレスを示す．

7.2 IPアドレス

1）アドレスの形式

インターネット上のホストやルータはインターフェースごとにIPアドレスをもつ．ホストやルータは自分のもつIPアドレス宛てのデータグラムを受信した場合には，上位レイヤにそれを渡す．それ以外の場合には，他のホスト宛てのデータグラムと考え，中継処理を行う．IPアドレスは，インターネット内でそれぞれのネットワークを一意に識別するためのネットワークアドレス部と，ネットワーク内でホストを一意に識別するためのホストアドレス部から構成される．これを**図13**に示す．アドレスを表記する場合には，各バイトを10進数で表現し，各バイトの間をドットで区切る．たとえば，10.5.3.15というふうにである．

IPアドレスは，当初は，クラスAからEまでの5種類に分かれていた．クラスによってアドレス部やホスト部の長さが異なり，アドレスを使用する組織のネットワーク規模に応じたクラ

〔図12〕IPヘッダ

New Products ― T-Engine用IPv6対応ミドルウェア「KASAGO for T-Engine」
（株）エルミックシステムは，トロンプロジェクトの標準開発プラットホーム「T-Engine」上で動作するTCP/IP（IPv4/IPv6）プロトコルスタック「KASAGO for T-Engine」を開発した．この製品は，パーソナルメディア（株）を通じて発売される．

スのネットワークアドレスを割り当てていた．**表4**にIPアドレスのクラスを示す．アドレスのクラスは，アドレスの最上位からのビットパターンで決まる．たとえばクラスAの場合には，最上位ビットが0である．クラスBの場合には，最上位の2ビットが10のパターンである．

しかし，クラスAなどは，収容可能ホスト数が16,777,216台にもなってしまう．こんなに多数のホストをもつ組織はありえない．つまり，ほとんど使用されないアドレスとなってしまう．このような使い方をすると，有限な資源であるIPアドレスの枯渇が心配される．そこで，アドレス領域を可変長とすることによって，各組織にちょうど良い数のホストが収容できるネットワークアドレスを割り当てることにした．これをクラスレスアドレスと呼ぶ．また，以前のクラス付きアドレスをクラスフルアドレスと呼ぶ．

2）クラスレスアドレスとネットマスク

クラスフルアドレスの場合，**表4**に示すように，そのアドレスの上位ビットのパターンを見ただけで，どこまでがネットワークアドレス部で，どこからホストアドレス部かがわかる．しかし，クラスレスアドレスの場合には，そうもいかない．そこで，ネットマスクというものを使用する．

ネットマスクはネットワークアドレス部として使用する部分のビットを1にし，ホストアドレス部として使用する部分を0にした32ビット長のビット列である．これは，アドレスと同じく，各バイトを10進数で表記し，それらの間をドットで区切る．

たとえば，クラスAのネットワークアドレス10.0.0.0は，ネットマスク255.255.255.0を使用すれば，10.0.0から10.255.255までの約6万4千個のクラスC相当のネットワークアドレスとして使用できる．あるホストに10.5.3.15というアドレスをつけたとする．その場合，アドレスとマスクとのAND演算をすると10.5.3というネットワークアドレス部の値が得られる．その残りである15がホストアドレス部である．ネットマスクの表記方法には，スラッシュの後にマスクの長さを示す十進数を書いて示す方法もある．たとえば，/24は，255.255.255.0と等価である．

7.3 経路制御（Routing）

1）経路テーブル

ネットワークレイヤの仕事は，IPデータグラムを転送先のホストまで運ぶことである．転送元のホストでは，トランスポートレイヤから渡されたセグメントをIPデータグラムのデータ部に入れ，指定された転送先アドレスを設定する．次に，その転送先アドレスを見て，次にこのデータグラムを渡すべき隣接したルータを決定する．その決定は，経路テーブル（Routing Table）に基づいて行われる．このテーブルには，転送先のIPネットワークアドレス，ネットマスク，転送先までの経路上にある隣接したルータのIPアドレス，そのルータが接続されているインターフェースの番号などが入っている．検索の際には，転送先のIPアドレスに各エントリのネットマスクをかけ，IPアドレスが一致するものをさがす．ホストやルータは，一致したエントリ中のインターフェースを通じて次のルータへパケットを転送する．

表5に経路テーブルの例を示す．たとえば，172.16.5.3へのデータグラムを受信した場合には，172.16.5.0/27が一致する．このため，次の192.168.32.15に渡さなければならないことがわかる．その際には，番号2のインターフェースから送り出す．

表5でIPアドレスの部分にdefaultと書かれているエントリは，検索がどのエントリにもヒットしない場合に使用するエントリである．すなわち，送り先のわからないデータグラムはすべてこのエントリのルータに送る．これは，経路の知識が少ないルータやホストは，他のもっと知識のあるルータに経路制御をまかせればよいということである．これをデフォルト経路（default route）と呼ぶ．なお，以上は，中継の基本的原理を示したものであり，実際には，高速化のためのさまざまな手法が使用されている．

ホストや末端のルータは，デフォルト経路を使用すれば，設定が簡単になる．しかし，インターネットバックボーンのルータはそういうわけにはいかない．それらは，すべてのIPアドレス

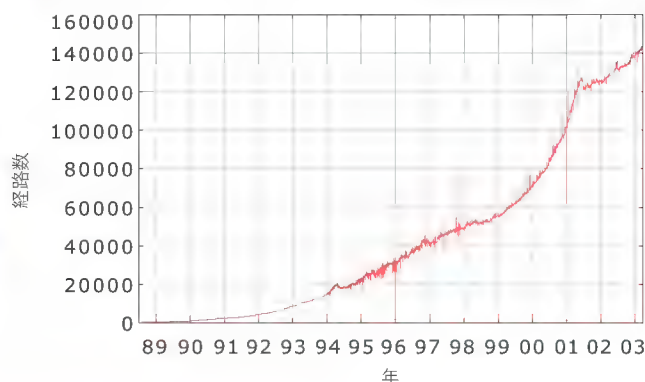
〔表4〕IPアドレスのクラス

クラス	上位ビットパターン	ネットワークアドレス範囲	ホストアドレス部の長さ	備　考
A	0xxx	0.0.0.0-127.0.0.0	3バイト	ユニキャスト
B	10xx	128.0.0.0-191.255.0.0	2バイト	ユニキャスト
C	110x	192.0.0.0-223.225.225.0	1バイト	ユニキャスト
D	1110	224.0.0.0-239.255.255.255	なし	マルチキャスト
E	1111	240.0.0.0-255.255.255.255	なし	予約

〔表5〕経路テーブルの例

IPアドレス	ネットマスク	次のルータのアドレス	次のルータが接続されているインターフェース番号
10.1.1.0	/26	192.168.15.31	1
192.168.1.0	/24	192.168.15.52	1
172.16.5.0	/27	192.168.32.1	2
default	―	192.168.32.15	2

〔図14〕インターネットバックボーンにおける経路数の推移
（http://bgp.potaroo.net/as1221/bgp-active.htmlより引用）

New Products ― Xtensaコアを使用した高性能MPEG-2 CODEC ASIC
テンシリカ社は，NTTエレクトロニクス(株)(NEL)が米国テンシリカのXtensaマイクロプロセッサコアを搭載したCODEC 2品種の出荷を始めたことを発表した．「NEL VASA」チップと「NEL SuperENCIII」チップで，高性能な高品位TVから低コストの双方向ビデオ通信システムにいたる用途向け．

に関する経路を経路テーブルにもつ．このため，経路テーブルが非常に大きい．その経路数の推移を，**図14**に示す．1994年には2万経路程度だったものが，2003年には14万経路を越えている．

2）経路制御プロトコル

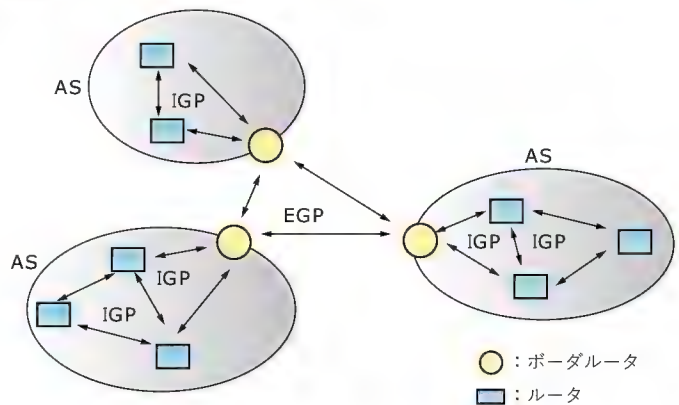
経路テーブルを自動的に作成するために，経路制御プロトコルがある．これは，アプリケーションレイヤのプロトコルではあるが，ネットワークレイヤに関連が深いのでここで説明しておく．隣接するルータは経路情報を定期的あるいは経路に変化のあったときに交換する．そして，隣接するルータのもつ経路情報を学習し，自分のもつ経路情報と学習した経路情報の中から最短距離の経路を選択する．これが，さらに隣接するルータに通知される．このような経路情報の交換と最短経路の計算がネットワーク中のルータで行われ，結局，各ルータはネットワーク中のそれぞれのネットワークアドレスに関して最適な経路を得る．このような経路制御はAS内と，AS間とで独立して行われる．

前者のためのプロトコルにはいくつかあるが，それらを総称してIGP（Interior Gateway Protocol）と呼ぶ．また，後者のためのプロトコルを総称してEGP（Exterior Gateway Protocol）と呼ぶ．このようすを**図15**に示す．

IGPの代表として，RIP（Routing Information Protocol）やOSPF（Open Shortest Path First）プロトコルがある．RIPはRFC1058で規定されており，最短経路を選択する基準として，目的のネットワークまでの距離を使用する．これを距離ベクタ型プロトコルと呼ぶ．経路情報を受信した場合には，最短経路の計算結果を経路テーブルに反映させ，さらに，それを隣接するすべてのルータへブロードキャストする．

一方，OSPFはRFC2328で規定されており，各ルータが自分と隣接するルータとの接続地図を，ネットワーク内のすべてのルータに通知する．そして，各ルータが独立して接続情報から最短距離の経路を計算する．これをリンクステート型プロトコルと呼ぶ．これらの経路情報の交換は動的に行われるため，ネットワークに障害が発生した場合には経路テーブルが自動的に変更され，その経路を避けて別の経路が使われる．

EGPの代表としてBGP（Border Gateway Protocol）がある．これは，RFC1771などで規定されている．BGPでは，各ASの入口のルータ（これをボーダルータと呼ぶ）が，AS内にあるネットワークのアドレスを他のASのボーダルータと交換する．その情報には，ネットワークアドレスだけでなく，そのアドレスが所属するASを識別するためのAS番号やネットマスクが入れられる．この場合，AS配下のクラスレスアドレスはまとめられる．

たとえば，209.185.0/17と209.185.128/17のネットワークがAS配下にあった場合，これらは経路情報209.185/16として通知される．このおかげで，経路情報の数を抑制でき，バックボーンルータでのメモリ不足による経路情報のあふれを避けることができる．これをサイダーCIDR（Class-less Inter-Domain Routing）と呼ぶ．なお，インターネットでは，ルータのことをかつてゲートウェイと呼んでいた．このために，ゲートウェイという言葉が経路制御プロトコルの名称に残っている．

〔図15〕IGPとEGP

経路制御をさらに詳しく調べたい読者は，参考文献4）を参照していただきたい．また，実際のデータグラムの転送経路を知るツールが提供されているので，興味のある方は，調べてみると経路制御の実感がわく．たとえば，UNIX系OSには`traceroute`コマンドが，Windows系OSには`tracert`コマンドがある．これらは，手元のホストから目的のネットワークアドレスへの経路を調べることができる．しかし，別のホストからの経路を調べることはできない．そのようなときには，`traceroute`を実行させてくれるサービスを利用する．

たとえば，`http://www.traceroute.org/`には，`traceroute`をWebから実行することのできる世界中のホストが一覧表にまとめられている．また，経路テーブルを見たい場合には，UNIX系OSやWindows系OSには`netstat -r`コマンドがある．各社のルータにも，同様の機能のコマンドがある．

7.4 アドレス解決

ホストやルータがIPデータグラムを転送する場合には，それを転送するリンク固有のフレームにエンキャプシュレーションする．それが専用線であれば，接続された相手は決まっているので，フレームを転送するだけでよい．しかし，Ethernetのように多重アクセスができ，多数のホストが同一ネットワークセグメント上に接続されている場合には，どのホスト/ルータに渡すかをフレームの転送先アドレスで指定する必要がある．

たとえばEthernetの場合には，転送元と転送先のMACアドレスをフレームヘッダに設定する．前者は，自分のアドレスであるので既知である．しかし，後者に関しては調べる必要がある．転送先のホストが，同一ネットワークセグメント上にある場合には，そのIPアドレスに対応したMACアドレスを知る必要がある．

また，別のネットワークセグメント上にあって，同一ネットワークセグメント上にあるルータの一つに渡さなければならない

New Products — TIの720MHz動作のDSP
日本テキサス・インスツルメンツ（株）は，720MHz動作のDSP 3製品「TMS320C6416」「TMS320C6415」「TMS320C6414」を発売した．64チャネルのEDMA，128個のTDM（時分割多重）チャネルをサポートするMcBSP 3組などが特徴．価格は$199.89～（10,000個時）．

場合には，経路テーブルでそのルータのIPアドレスを検索し，そのIPアドレスに対応するMACアドレスを調べなければならない．この対応を動的に見つけるためにARP(Address Resolution Protocol)を使用する．これは，IPアドレスからMACアドレスを調べるプロトコルであり，リンクレイヤとネットワークレイヤの両方に関連する．リンクによってアドレスの構造が異なるので，ARPもリンク種別に対応して用意されている．たとえばEthernetの場合には，RFC 826である．

ARPでは，まず，MACアドレスを調べたいホストのIPアドレスを入れたARP要求パケットをブロードキャストフレームで転送する．その転送先のMACアドレスはブロードキャストを示すオール1の値である．これは，ネットワークセグメント上のすべてのホストやルータで受信される．それらのホストやルータでは，それに含まれていたIPアドレスをチェックし，自分のアドレスと一致した場合には，ARP応答パケットを返す．その中にはMACアドレスが入っている．これにより，ARP要求を送信したホストやルータは，所望のMACアドレスを得ることができる．

このようすを**図16**に示す．この図ではホストAからホストDまでの4台が同一ネットワークセグメントに接続されている．そして，ホストDが10.0.0.3へのARP要求をブロードキャストし，ホストCがMACアドレス00-50-da-b7-ee-2dをARP応答パケットでホストDに返している．

いったん受信したARP応答パケットの情報は，しばらくテーブルにキャッシュされている．したがって，それがクリアされるまでの間は，ARP要求を出さずにアドレスの対応を知ることができる．UNIX系OSやWindows系OSには，このテーブルを表示するコマンドが用意されている．前者の場合にはarp，後者の場合にはarp -aを実行するとテーブルの内容が表示される．

〔図16〕ARPの動作

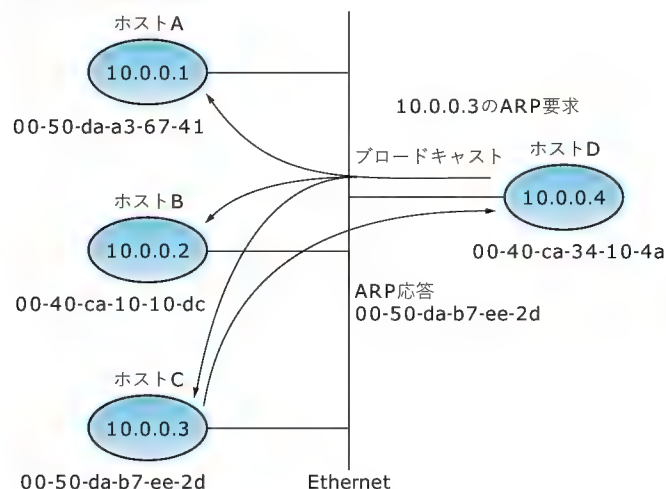
8 トランスポートレイヤのプロトコル

8.1 ポート番号

トランスポートレイヤにはTCPとUDPとがある．これは，エンドノード，すなわち，ホスト上で動作する．TCPは，転送するデータに対する応答確認を行う．送信側のホストは，転送先のホストから受信したという旨の応答が返ってこないデータがあれば，転送中にパケットが損傷したり失われたと考え，再転送を行う．一方，UDPは，応答確認を行わず，再転送によるエラーからの回復を行わない．このため，ホストの処理はきわめて軽いものとなる．UDPを使用して，しかも信頼性を必要とする場合には，アプリケーション側で，TCPと同様の処理を行わなければならない．

これらのプロトコルの重要な仕事の一つは，アプリケーションのプロセスごとに独立した通信路，すなわち，バーチャルサーキットを提供することである．通常，バーチャルサーキットはエラーのない通信路のことをいうので，UDPのほうはバーチャルサーキットとは呼ばないが，ここでは説明の便宜上，同様に呼ぶことにする．各セグメントがどのバーチャルサーキット上のものかを識別するため，TCPやUDPのヘッダには，転送元ポート番号と転送先ポート番号とが入れられている．各ホストでは，このポート番号の組，および通信先のIPアドレスで，それがどのバーチャルサーキットに属するものかを識別する[注2]．

すなわち，パケットの受信の際には，これらを参照してそれぞれのバーチャルサーキットに対応したバッファに振り分ける．このテーブルのようすは，UNIX系やWindows系OSの場合には，netstat -anコマンドで見ることができる．このコマンドでは，それぞれのバーチャルサーキットの状態なども表示される．また，ポート番号は192.168.1.4.23のように，IPアドレスの後にドットを付けて連続して表示される．バーチャルサーキットの識別には，ポート番号の組と通信先のIPアドレスに加え，自IPアドレスも使用されている．

通信要求を受理するサーバ側のポート番号は，アプリケーシ

〔表6〕代表的な既定のポート

サービス	TCPポート番号(十進)	UDPポート番号(十進)
ECHO	7	7
FTP	21	—
Telnet	23	—
SMTP	25	—
HTTP	80	—
DNS	53	53

注2：この二つで識別できるが，転送元のIPアドレスを加えてもかまわない．転送元のIPアドレスは，相手のホストへパケットを送り出すインターフェースごとに異なる．

New Products — 4バンドダイレクトコンバージョンGSM/GPRS送受信機
アナログ・デバイセズ社は，ダイレクトコンバージョン無線装置Othelloファミリの新製品「OthelloOne TV」を発売した．これまでのOthello送受信機のすべての機能に加え，従来は外付けしていた送信用VCOを内蔵している．

〔図17〕UDPセグメント形式

UDPヘッダ	UDPデータ

〔図18〕UDPヘッダ

0	16 31
Source Port	Destination Port
Length	Checksum

〔図19〕疑似ヘッダ

0	8	16 31
Source IP Address		
Destination IP Address		
Padding	Protocol	Length

〔図20〕TCPセグメント形式

TCPヘッダ	TCPデータ

〔図21〕TCPヘッダ

0	4	8	16 31
Source Port			Destination Port
Sequence Number			
Acknowledge Number			
Data Offset	Reserved	Code Bit	Window
Checksum			Urgent Pointer
Options			Padding

ョンのサービスごとに決められている．また，通信要求を送るクライアント側は，このポート番号を指定してバーチャルサーキットを設定するためのコネクションの確立要求を出す(8.3を参照)．その際の，転送元ポートの番号は，自ホスト内ですでに使用しているポート番号と衝突しないように決められる．**表6**にTCPとUDPの代表的なポートの番号を示す．これらは既定のポート(well-known port)と呼ばれている．ポート番号の1から1023番までは，この既定のポートのために予約されている．動的に使用できる番号は1024から65535番までである．

8.2　UDP

UDPはRFC768で規定されている．このセグメントの形式を**図17**に示す．これは，ヘッダ部とデータ部とから構成される．データ部には，アプリケーションから渡されたデータが入る．ヘッダ部を詳しくしたものが**図18**である．**図18**では，パケットの先頭を左上に書いており，転送は上から下へ行われる．また，32ビット幅でフィールドを示している．

ヘッダは，転送元ポート(Source Port)と転送先ポート(Destination Port)から始まる．長さ(Length)フィールドには，UDPセグメント全体の長さがバイト単位で入れられる．また，チェックサム(Checksum)フィールドは，受信側でセグメントの誤りが検出できるようにするものである．これは，次のように計算した値を入れる．まず，**図19**の疑似ヘッダを作成する．これは実際に転送されるものではなく，チェックサムの計算のためだけに使用される．埋め草(Padding)フィールドには0が入る．また，プロトコル(Protocol)フィールドにはUDPのプロトコル番号である17(バイナリで00010001)が入る．長さ(Length)フィールドには，UDPのセグメント長が入る．

この仮想ヘッダをUDPセグメントの前につける．さらに，疑似ヘッダとセグメントの合計の長さが16ビットの倍数になるようにセグメントの最後に0を付加する．また，ヘッダのチェックサムフィールドは0にしておく．そして，これに対して16ビットごとの“1の補数の和”の“1の補数”[3]を求める．これをチェックサムフィールドに入れる．なお，計算結果が0であれば，それをオール1にする．また，受信側でチェックサムを使用しない場合には，オール0にして送る．

受信側では，送信側と同様の計算を行う．すなわち，IPヘッダからUDP疑似ヘッダを作成して，チェックサムを再計算する．その結果が0になれば，このセグメントは損傷を受けていない正しいものということになる．

8.3　TCP

1）セグメントの形式

TCPは，RFC793で規定されている．このセグメントの形式を**図20**に示す．データ部には，アプリケーションから渡されたデータが入る．**図21**は，詳しいヘッダ部のフィールドである．これは，パケットの先頭を左上に書いており，転送は上から下へ行われる．また，32ビット幅でフィールドを示している．

ヘッダは，転送元ポート(Source Port)と転送先ポート(Destination Port)から始まる．順序番号(Sequence Number)，応答確認番号(Acknowledge Number)，ウィンドウ(Window)，フラグを入れるコードビット(Code Bit)については後述する．データオフセット(Data Offset)は，4バイトを単位としたTCPのヘッダ長を示す．これを見れば，どこからTCPのデータ部が始まるかがわかる．オプションがない場合にはTCPのヘッダ長は20バイトなので，そのオフセットは5となる．チェックサム(Checksum)フィールドは，UDPの場合と同様に受信側でセグメントの誤りが検出できるようにするものである．この作成や検査は，UDPと同じようにして疑似ヘッダを使用して行われる．その際，プロトコル(Protocol)フィールドにはTCPのプロトコ

Information ― シスコとIBM，シスコSANスイッチ製品の日本での再販で協業
シスコシステムズ(株)と日本アイ・ビー・エム(株)は，シスコのSAN(ストレージ・エリア・ネットワーク)対応のスイッチ/ダイレクタ製品「Cisco MDS 9000」ファミリを日本IBMおよび同社のビジネスパートナ企業が再販していくことを発表した．

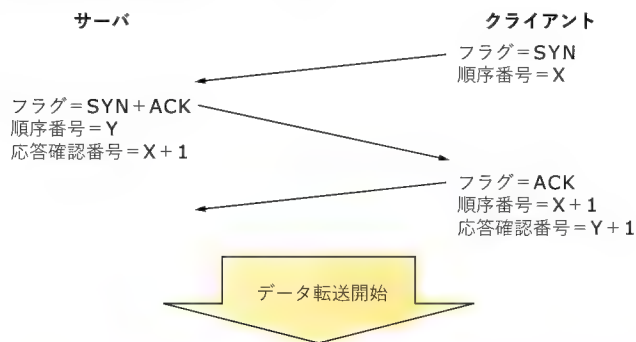
ル番号6(バイナリで0000 0110)を入れる．これはIPヘッダのプロトコルフィールドと同じ値である．また，長さ(Length)フィールドには，TCPのセグメント長を入れる．

2）コネクションの確立

TCPでは，通信の前にバーチャルサーキットを張る．これを，コネクションの確立と呼ぶ．通信終了時にはコネクションの解放を行う．これには，ヘッダ中のコードビット，順序番号，応答確認番号を使用する．コードビットとは6ビット長のフラグで，それぞれのビットに意味が与えられている．このうち，Synchronize Flag(SYNフラグ)，Acknowledgement Flag(ACKフラグ)，Fin Flag(FINフラグ)がコネクションの確立や解放に使用される．順序番号とは転送するデータのバイトごとに付けられた番号である．それを受理した側では，受理した次のデータの順序番号を応答確認番号として返す．すなわち，次に受信を期待する番号ということになる．

i）まず，接続要求を行うクライアントは，サーバに対し，SYNフラグを立てたセグメントを送信する．その順序番号フィールドには，このフラグに付けられた順序番号が入っている．すなわち，フラグも1バイトのデータとみなされる(ただし，ACKフラグは0バイト相当である)．また，転送先ポート番号はサーバが提供する特定のサービスを示す番号である．**表6**にすでに示したように，80ならばHTTPのサービスを示す．転送元ポート番号は，既定のポート以外であり，しかも，自ホスト中で他のTCPのバーチャルサーキットが使用していないものである．なお，このACKフラグは0であり，応答確認番号が意味をもたないことを示す．

ii）サーバは接続要求を受理すると応答を返す．そのTCPヘッダには，SYNとACKフラグが立てられている．このACKは受理したSYNに対する応答確認であり，その応答確認番号は受信した順序番号＋1の値である．また，順序番号は，送信するSYNフラグに付与された番号である．

iii）これを受信したクライアントは，サーバ側からのSYNフラグに対する応答確認を返す．このTCPセグメントのヘッダには，ACKフラグだけが立っている．そして，その応答確認番号はサーバからのSYNに対するものでサーバ側からの順序番号＋1の値である．

以上の3ステップでコネクションが確立され，バーチャルサーキットが張れたことになる．このスリーウェイハンドシェイク(3way handshake)を**図22**に示す．データの転送と同じく，応答が期待する時間以内に受信できない場合には，セグメントの再送信を行う．

3）TCPにおけるコネクションの解放

データ転送が終了し，コネクションを解放する場合には，FINフラグを立てたセグメントを送る．これは，サーバ側から行っても，クライアント側から行ってもよい．それを受信した側では，FINも1バイトのデータとみなすので，FINに対する確認応答を返す．このとき，すぐに両方向の通信を停止してコネクションの解放を行う場合には，FINフラグを立てておく(まだ送るべきデータがある場合には，FINに対する確認応答はするが，そのまま転送を続けるのでFINフラグを立てないセグメントを送る)．これに対して，最初にFINを送った側は，FINに対する確認応答を返す．これで，コネクションの解放が完了する．

このようすを**図23**に示す．この図では，ホストB側からの解放要求に対してホストA側もすぐに解放を行う例を示している．データの転送と同じく，応答が期待する時間以内に受信できない場合には，セグメントの再送信を行う．

4）TCPのデータ転送とウィンドウ制御

TCPにおけるデータ転送では，それぞれのデータのバイトに順序番号がつけられ，受信側では，次に受信を期待する順序番号(すなわち，それ以前のバイトは受理したということを意味する)を応答確認番号として返す．その際，ヘッダ内の応答確認番号が有効であることを示すためにACKフラグを立てておく．送信側では，転送したデータに対する応答確認が期待する時間以内に戻ってこない場合には，セグメントの損失などによってエラーが発生したと考え，再転送を行う．

応答確認を返す場合には，ウィンドウサイズ(Window Size)を入れておく．これは，応答確認番号に相当するデータも含めて受信可能なバイト数を示す．通常は，空いている受信バッファの量をこのウィンドウサイズに指定する．たとえば，応答確認番号が1000で，ウィンドウサイズが4096であれば，1000から5095までの順序番号をもつデータの送信を許可する意味になる．

〔図22〕スリーウェイハンドシェイク

サーバ　　クライアント

フラグ＝SYN
順序番号＝X

フラグ＝SYN＋ACK
順序番号＝Y
応答確認番号＝X＋1

フラグ＝ACK
順序番号＝X＋1
応答確認番号＝Y＋1

データ転送開始

〔図23〕コネクションの解放

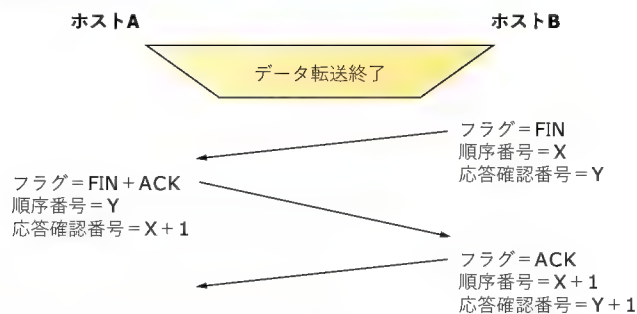
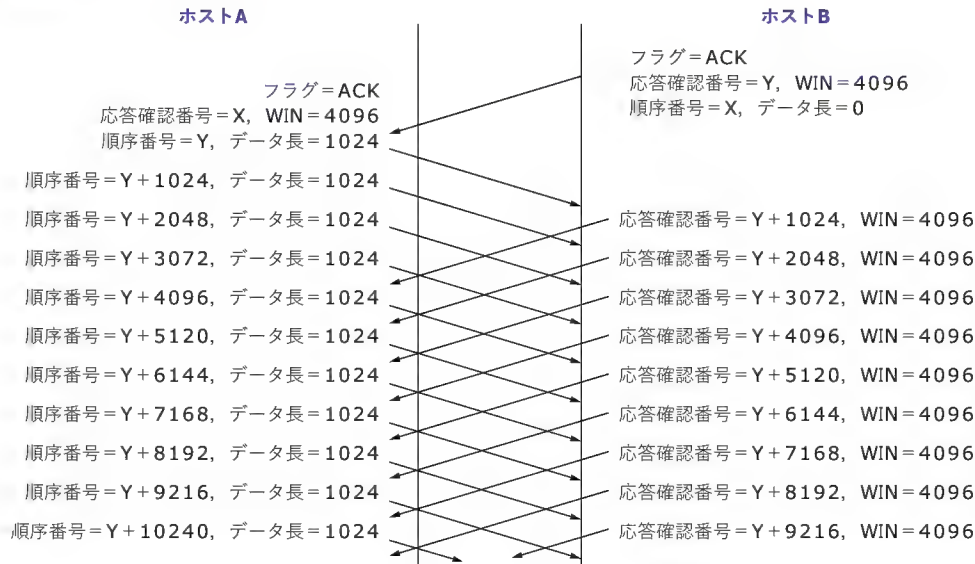
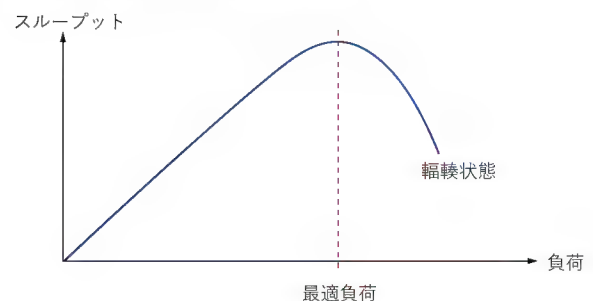
New Products ― 組み込みLinuxリファレンスターゲットスタータキット
(株)イーエルティは，MontaVista Linuxを市販のリファレンスボード上で評価できる「組み込みLinuxリファレンスターゲット・スタータキット」を開発した．この製品は，各種のリファレンスボードでMontaVista Linuxを容易に実装できる形で提供するもの．

〔図24〕ウィンドウの制御

送信側は，このデータ量以内であれば，応答確認を待たなくても複数のセグメントを転送することができる．このウィンドウ制御によって，転送のスループットが向上する．もしも，相手からの応答確認を必ず待って次のセグメントを転送しなければならないとすると，相手へデータを送って応答確認が返ってくる往復時間（これをラウンドトリップタイムという）は転送が停止する．とくに，回線の速度が遅い場合や，遠距離の通信でネットワークの遅延が大きい場合[注3]には，転送できる時間よりも待ちの時間のほうが，多くスループットが著しく低下する．

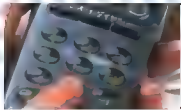
図24にはウィンドウを使用したデータ転送のようすを示す．この例では，ホストAからBへデータを転送している．ホストBが許すウィンドウサイズ（WIN）は4096である（ここでは，説明を簡単にするため，常に同じサイズとしている）．このため，ホストAは，最初に四つまでの1024バイト長のデータを入れたセグメントを連続して転送する．四つ目を転送したときに許可されたウィンドウ（最後のデータの順序番号はY＋4096－1）を使い切り，待ちに入るが，その直後に，最初に送ったセグメントに対する応答確認が届くので，順序番号でいうとY＋1024＋4096－1までの転送ができるようになる．そこで次のセグメントを送るが，ウィンドウを使い切り待ちに入る．その直後に2番目に転送したセグメントに対する応答確認は届くので，1024バイトのデータが転送できるようになる．このようにして，応答確認の転送とデータ転送とが重畳され，ラウンドトリップの遅延が大きくても効率の良い転送が行われる．

5）輻輳ウィンドウ

受信したウィンドウサイズは，相手のホストがどれだけ次のデータを受理できるかを示す．これにはネットワークの混雑状態は考慮されていない．そのため，相手が許可したとおりのデータ量を転送した場合，ネットワークを過負荷に陥らせる可能性がある．また，ネットワーク内の混雑でデータグラムが失われた場合，むやみに再転送を行うと，さらにそれが悪化し，ネットワークが輻輳（過負荷）状態に陥る場合がある．しかし，インターネットはブレーキのない車と同じで，それ自身にはフロー制御がないので，末端のホストが混雑状態を推測して転送量や転送のタイミングを制御しなければならない．これが長年，インターネットプロトコルを開発する研究者を悩ませてきた．実際，フロー制御の失敗で，インターネットは過去何度も輻輳崩壊（Congestion Collapse）という状態に陥った．ネットワークを過負荷にしないようにしながら可能なかぎりの最大の転送スループットを得る最適制御は，至難の業である（**図25**）．なぜならば，ネットワークは膨大な数のユーザーによって使用され，刻々と負荷が変動するからである．

〔図25〕スループットと負荷の関係

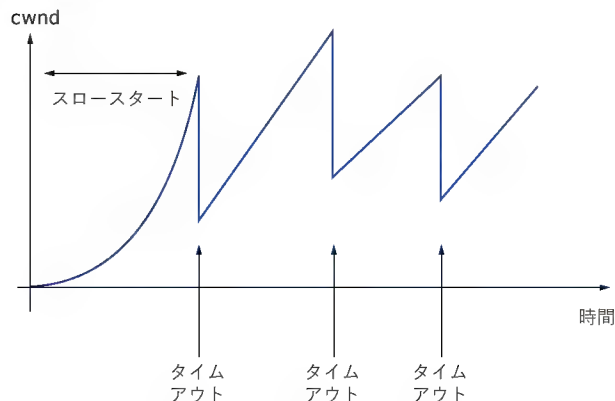
そこで，いろいろな方法が研究されてきた．まず，パケット

注3：遅延と帯域（速度）の積をdelay bandwidth productと呼び，これが大きいほうがスループットが低下する．

New Products ―リネオ，東芝アメリカと提携しAVM79BOX対応の組み込みLinuxを提供
リネオソリューションズ（株）は，Toshiba America Electronic Components（TAEC）と提携し，TAEC開発のAVM79BOXに対応する組み込みLinuxを提供することを発表した．AVM79BOXは，TX7901 MIPS RISCコアを搭載したメディアセンター向けの評価ボードである．

の損失をネットワークからの輻輳の警告だと考え，ウィンドウサイズを絞るという方法である．これは，輻輳ウィンドウcwnd（Congestion Window）というホストの内部処理のためだけのウィンドウをもつことによって実現されている．もし相手からのウィンドウサイズよりもこのcwndが小さい場合には，これにしたがう．逆の場合には，相手のウィンドウサイズにしたがう．パケットの損失（確認応答が既定時間内に返ってこない＝タイムアウトで検出）が検出された場合には，cwndをそのときのサイズの半分にする．

しかし，そのままでは転送速度が落ちたままである．そこで，確認応答を受信するたびにcwndを1/cwndだけ増加させる．ただし，転送のスタート直後だけは，1セグメント分から輻輳ウィンドウを始め，相手から確認応答を受信するごとに1セグメント分を増加させる．これをスロースタートと呼ぶ．それによって，今度は確認応答も2倍のセグメント分になるので，結局cwndは指数関数的に増加する．このようすを**図26**に示す．TCPの実装によっては，このような配慮がまったくなされておらず，単に再転送をくり返すものがある．そのような実装は，インターネットにとって脅威となる．また，UDPのほうは，このようなフロー制御がアプリケーションにまかされている．こちらのほうも，転送アルゴリズムによってはインターネットの輻輳崩壊を発生させる可能性がある．

輻輳ウィンドウによる転送量の制御だけでなく，再転送のタイミングの制御も重要である．再転送の判断を行うタイマ値の決定は，次のようにネットワークの状態によって動的に制御される．すなわち，相手からの確認応答が返ってくる時間（ラウンドトリップタイム）を常に監視しておき，その平均値と分散をベースに計算する．ネットワークが混雑してくるとラウンドトリップは増加する．このため，タイムアウトを判定するまでのタイマの値も大きくなり，再転送を行うまでの時間が長くなる．

〔図26〕cwndの変化

RFC793には，ローパスフィルタを使用した計算方法が規定されている．しかし，平均の遅延しか考慮していないため，分散が大きくなる高負荷のネットワークでは過小評価によってさらにパケット再送が発生し，ネットワークを輻輳状態に陥らせる危険がある．そこで参考文献5）や参考文献6）では，遅延の平均と分散の両方を考慮した方法を示している．フロー制御に関しては，上記も含め，さまざまな解決すべき問題がある．興味のある方はRFC1122，2001，2581，2582，2861，2988，3042などを参照していただきたい．また，ウィンドウ制御全般については，参考文献7）を参考にしていただきたい．

6）パスMTUの発見とその問題

フラグメンテーションの抑制はネットワークレイヤに関連するが，TCPセグメントの最大データ長MSS（Message Segment Size）を利用して，データグラム長の調整を行っているので，ここで説明しておく．MSSはTCPセグメントのデータ部の最大長である．

フラグメンテーションはパケット転送の上からは望ましいものではない．なぜならば，そのうちのどれか一つでも欠落すると全部が送り直しになるからである．また，組み立てる際には全部のフラグメントがそろうまでバッファに保存しておく必要がある．さらに，フラグメントはバーストで送られる傾向があるので常にバッファオーバフローが発生し，後のほうにあるフラグメントが捨てられ，何度再転送しても通信できないという事態に陥ることがある．これらの問題を防止するためには，転送元のホストで，ネットワーク中でフラグメントが発生しないようにデータグラム長を制限すればよい[注4]．そのためには，転送元のホストが，転送先ホストまでの経路中にあるもっとも小さいMTUを知る必要がある．そこで，Path MTU discoveryという手法が開発された．これはRFC1191で規定されている．

この方法では，すべてのIPデータグラムのフラグ（Flag）部分に，フラグメンテーション禁止を指定するDF（Don't Fragment）ビットをセットして転送する（Flagのフィールドについては，**図12**を参照）．もしも，途中のルータでフラグメントが必要になった場合には，そのルータは分割ができないので，それを通知してくる．この通知には，ネットワーク状態の通知や管理などに使用するICMP（Internet Control Message Protocol）のメッセージが使用される．転送元ホストにそれが返ってきた場合には，そのホストは，TCPが一度にIPに渡すことのできる最大のセグメントサイズMSS（Message Segment Size）を小さく再設定する．そして，このICMPが返ってこなくなるまで，これを繰り返す（なお，オプションのないIPとTCPのヘッダを想定すると，ヘッダの合計が40バイトなので，MSSとパスMTUにはMSS＝パスMTU－40の関係がある）．

ところが，RFC2923で報告されているように，いくつかの問

注4：データグラムを送り出すリンクのMTUはリンク固有の値なので，通信する相手（パス）によって変更できない．このため，MSSを利用して最大データグラム長を制限する．

New Products ― マイクロプロセッサ用電源向け高電流CMOS LDOファミリ
ナショナルセミコンダクタージャパン（株）は，PC，マイクロプロセッサ，DSPなど低電圧・高電流アプリケーション向けの電源用に，低ノイズCMOS低ドロップアウト（LDO）スイッチングレギュレータの新ファミリ「LP387xファミリ」を発売した．価格は$1.15～（1,000個時）．

題がある．たとえば，むやみにICMPの生成や通過をフィルタしているISPのルータやファイアウォールが途中にあると，ICMPが返ってこない．このため，転送元のホストはIPデータグラム長がパス中の最小MTU以下であると判断する．しかし，DFフラグが立っているので，途中のルータで破棄される．これはブラックホールと呼ばれる．この問題は，通信する相手やパスによって通信できたりできなかったりするので始末が悪い．これを避けるにはPath MTU discoveryを使用せず，データグラムを送り出すインターフェースのMTUを変更して十分小さく設定しておく方法が取られる．

9 アプリケーションレイヤのプロトコル

9.1 代表的なアプリケーション

アプリケーションレイヤには，電子メールの転送プロトコルであるSMTPや，Webページの閲覧をするためのプロトコルであるHTTP，ファイル転送のためのFTP，遠隔のホストを手元の端末から使用するための仮想端末プロトコルTelnetなどがある．これらはTCPやUDPを使用する．アプリケーションのソフトは，サーバ-クライアントモデルでサービスを提供するものが多い．この場合，クライアント側からサーバ側へコネクションを設定し，そのバーチャルサーキット上でメッセージをやりとりして処理を進める．

アプリケーションの中には，FTPのように，制御用とデータ転送用に二つのバーチャルサーキットを使用するものもある．制御用のバーチャルサーキット上では，ユーザーをパスワードで認証したり，ファイル名や操作要求をやり取りする．また，データ転送用のバーチャルサーキットのポート番号を決めるためにも使う．データ転送用のバーチャルサーキットは，制御用のそれとは逆に，サーバ側からクライアント側へコネクションを設定する．

なお，クライアント側からサーバ側へこのコネクションの設定を行うオプションもある．これをパッシブモードと呼び，ファイアウォールや，アドレスを変換するNAT（Network Address Translation）のようにインターネットから企業や家庭内部のネットワークへ向かってコネクションの確立ができない装置を介した場合に使用する．前者は，SYNフラグだけのついたセグメントを破棄することによって外部からのアクセスを禁止している．また，後者はアドレスを変換するので，外部から見たアドレスと内部のホストのアドレスが異ってしまう．

UDPの場合には，TCPのようにコネクションの設定は必要なく，クライアントは指定したポートへセグメントをいきなり送る．UDPを利用する代表的なアプリケーションとしてDNS（Domain Name System）がある．DNSはホスト名をIPアドレスに変換したり，その逆を行ったり，メールの配送先を知るために使用する．たとえば，ブラウザでURLに`www.bar.com`を指定したとする．これは，DNSサーバへUDPで問い合わせが行われ，IPアドレスに変換される．そして，そのアドレスに対してHTTPのコネクションが張られる．また，`foo@bar.com`宛てのメールを書いた場合には，メールサーバは，DNSで`bar.com`宛てのメールの配送先を問い合わせる．そして，そのホスト名を知り，そのIPアドレスに対してSMTPのコネクションを張る．

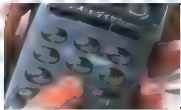
9.2 アスキー文字ベースのプロトコル

IETF文化のアプリケーションプロトコルで特徴的なのは，おもなものがTCPを使用し，アスキー文字（ASCII文字）ベースでメッセージができていることである．これで，診断やデバッグが簡単になる．たとえばメールシステムでは，Telnetを使用してサーバと会話すれば，メールが送れてしまう．以下でそれを示す．ここでは，SMTPサーバがfoo.bar.comというサーバで動いていると仮定する．まず，その25番ポート（すなわちSMTPポート）へTelnetで接続する．そして，手で以下の下線を引いた部分を入力する．これは，`user1@bar.com`から`user2@another.bar.com`へメールを送る例である．なお，頭が3桁の数字で始まる行はサーバからの応答である．これは，コマンド実行の結果を示す．

```
% telnet foo.bar.com. 25
Trying 111.111.111.111...
Connected to foo.bar.com.
Escape character is '^]'.
220 foo.bar.com ESMTP Sendmail 8.8.6 ……
helo bar.com
250 foo.bar.com Hello root@localhost,……
mail from: user1@bar.com
250 user1@bar.com... Sender ok
rcpt to: user2@another.bar.com
250 user2@another.bar.com... Recipient ok
data
354 Enter mail, end with "." on a line ……
Reply-To: user3@bar.com
Subject: Test Message

Hello
.
250 IAA13263 Message accepted for delivery
quit
221 foo.bar.com closing connection
Connection closed by foreign host.
%
```

Webサーバの場合も，Telnetを使用してHTTPの80番ポートへ接続すれば，閲覧することができる．もちろん，Webブラウザと違い，ファイルがそのまま出力されるので読むのには苦労する．

```
% telnet www.bar.com 80
Trying 111.111.111.123...
```


New Products ― TIの3種類のワイヤレス技術を統合した新型PDAコンセプトデザイン
日本テキサス・インスツルメンツ(株)は，ワイヤレスの音声通話とデータ接続を同時に提供する，無線LAN，Bluetooth，GSM/GPRSの3種類のワイヤレス技術を統合した業界初のPDAコンセプトデザイン，開発名「WANDA（Wireless Any Network Digital Assistant）」を発表した．

```
Connected to www.bar.com.
Escape character is '^]'.
GET /index.html
(ここでリターンを入れる)
<html>
<head>
<title> WWW.BAR.COM Home Page</title>
</head>
        :
(ここにindex.htmlのソースが表示される)
</html>

Connection closed by foreign host.
%
```

これは，IETF文化のVoIPに使用されるプロトコルSIP(Session Initiation Protocol)にも受けつがれている．一方，ITU-T文化のプロトコルのほうは，文字ではなくビット列にマッピングされたパケットの形式をしている．このため，こちらのほうは手で動かしてみるというわけにはいかない．

9.3 VoIPのプロトコル

1) 電話のシステム

本稿の最初に書いたように，インターネットを利用して電話のサービスを提供するプロトコルVoIPには，ITU-T文化のものとIETF文化のものがある．前者はH.323であり，後者はRFC3261のSIPである．ほかにITU-TとIETFが共同で開発したプロトコルRFC3015(megaco)/H.248もある．これは既存電話網との接続を行う大規模な分散型VoIPシステムの構築を行うためのプロトコルである〔これらの詳細については，参考文献8)や参考文献9)を参照のこと〕．ここではH.323とSIPを詳細に説明するのではなく，そのモデルを比較する．

H.323は，サービス品質が保証されないLAN上におけるテレビ会議音声などのマルチメディアの通信システムを実現するために開発され，1996年に勧告が出されている．その元となった，ISDNを使用してテレビ会議を行うH.320という規格もある．これらはゲートウェイを介して相互接続できるようになっている．

一方SIPは，1995年にIETFのmmusic(Multiparty Multimedia Session Control)WGで標準化が開始された．そして，RFC2543(後にRFC3261が出たため現在は旧版)が1999年に発行されている．現在では，標準化はmmusicからSIPやSIPPINGというWGに中心が移っている．

どのプロトコルにせよ，目的は電話のシステムを実現するということである．おおまかにいうと，電話のシステムは二つの機能から構成されている．まず，相手を呼び出して通話路を設定したり通話後に通信路を解放したりする呼制御(こせいぎょと読む．呼制御のことをシグナリングとも呼ぶ)，そして，符号化した音声の転送である．従来の電話でいうと，ダイヤルを回して相手を呼んだり受話器を置いたりするのが前者，受話器と相手の受話器との間で音声を伝えるのが後者である．

2) 音声の転送

H.323とSIPのプロトコル構成をそれぞれ**図27**と**図28**に示す．注意しなければならないのは，H.323のほうはさまざまなプロトコルの集合から構成されるプロトコル群であり，SIPのほうはVoIPシステム中の一部(呼制御)のプロトコルという点である．後者は，既存のプロトコルと組み合わせてVoIPのシステムが構成される．

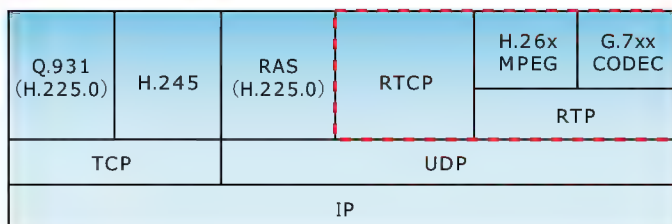
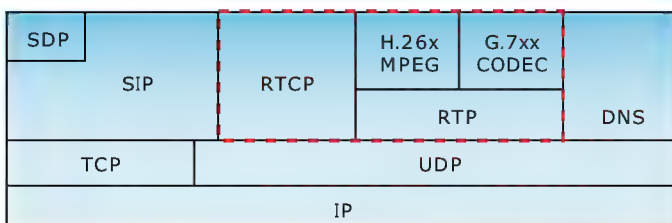
どちらも音声や動画の符号化と転送については似ている．UDP上でRFC1889のRTP(Real Time Protocol)およびRTCP(RTP Control Protocol)を使用し，その上でH.261(MPEG-1)，H.263(MPEG-4)などの動画符号化やG.711(64kbpsのPCM方式)，G.722(48/56/64kbpsの7KHz SB-ADPCM方式)，G.723.1(5.3/6.3kbpsのMP-MLQ/ACELP方式)，G.728(16kbpsのLD-CELP方式)，G.729(8/13kbpsのCS-ACELP方式)などの符号化を行った音声データを転送する．

RTPではパケットロス，逆転，重複の検出のために順序番号を使用する．また，パケット到着のゆらぎを補償して既定のタイミングで再生するためにタイムスタンプを使用する．タイムスタンプは，RTPデータの先頭のデータをサンプリングしたときのクロック値を示す．精度は，RTPデータの符号化方式に依存する．ゆらぎの補償のようすを**図29**に示す．送信側のホストで内部クロックによって10から90までのタイムスタンプを付けられたRTPパケットは等間隔で送り出される．しかし，受信したときには，ネットワークの遅延のゆらぎなどによって，パケットの間隔が送信時とは異なる．そこで，受信側では，タイムスタンプを参照し，自ホスト内のクロックを参照しながら，元のタイミングで再生する．RTCPはパケットロスやゆらぎ，転送バイト/パケット数などの品質や統計情報を送信側にフィードバックする．

〔図27〕H.323

〔図28〕SIP

細かい部分は一部省略した

New Products ── アルテラ，240個のIPコアをStratix，Cyclone FPGA製品に最適化
アルテラ・コーポレーションは，同社のStratixおよびCyclone FPGA製品向けに最適化された240個以上のIPコアを提供開始した．これらのIPコアは，すべてアルテラのOpenCoreおよびOpenCore Plus評価プログラムにより，購入前に無料で評価することができる．

3）呼制御

呼制御のほうは，H.323とSIPではスタイルが異なる．前者では，ISDNの呼制御で使用されているQ.931というプロトコルがベースとなる．これは，H.323を構成するプロトコルの中の一つ，H.225と呼ばれるプロトコルの一部である．H.225には，電話番号（別名）や通話許可に関する指定をデータベースに登録するためのプロトコルRAS（Registration，Admission，and Status）も含まれる．音声にどの符号化を使用するかのネゴシエーションは，H.245プロトコルで行われる．電話をかけると，H.323では，①サーバ（ゲートキーパ）に通信許可をH.225（RAS）で申請（UDPを使用）し，その返事に含まれているアドレスに②H.225（Q.931）で相手を呼び出し（TCPを使用），それが受理されれば，③H.245の手順で符号化方式を選択する（TCPを使用）という3ステップとなる．その後，会話が可能となる．このようすを**図30（a）**に簡単に示す．

一方，IETF文化のほうは，SIPというプロトコルが規定されている．これは例によって，アスキー文字でコマンドを送るプロトコルとなっている．これには，UDPでもTCPでも使用できる．SIPの原型はもともと，会議システムが元である．このシステムでは，`INVITE`というメッセージで会議への参加を打診し，打診された人がそれを受理するのであればOKの応答を返すというモデルを基本としている．SIPの場合，電話番号は，メールのあて先と同じ`user@sample.com`という形式にできる．この形式をURI（Uniform Resource Identifier）と呼ぶ．これをあらかじめ`REGISTER`メッセージでSIPのサーバ（SIPプロキシ）に登録しておく．`INVITE`メッセージは，そのサーバに送られ，そのサーバから登録されたホストへ`INVITE`メッセージが送られる．このようすを**図30**の（**b**）に示す．なお，`INVITE`を中継するのではなく，リダイレクト先のURIを教えるだけのリダイレクトサーバというのもある．この場合，リダイレクト先へ発呼先を変更して再度`INVITE`メッセージを送る．H.323のように音声符号化のネゴシエーションを別プロトコル（H.245）で行うのではなく，SIPでは，そのメッセージの中に，ネゴシエーションのためのメッセージを入れる．これは，SDP（Session Description Protocol）の規定にしたがう．このようなわけで，SIPのほうはプロトコルが簡単になっている．

なお，H.323もSIPも，ここで説明したものはほんの一部であり，これ以外の呼制御のモデルもあるので，詳しくは，以降の章を参照していただきたい．

おわりに

本稿ではまず，プロトコルの文化に言及し，今日のVoIPの基礎となるプロトコル群の背景を説明した．また，インターネットの設計理念とアーキテクチャから出発して，レイヤごとにTCP/IPプロトル群を眺めた．誌面の都合上，ドメイン名システムDNS，アドレス変換装置NAT，ファイアウォールなどの技術については説明を省略した．NATやファイアウォールを介して，透過的にVoIPのようなサービスを受けるには一工夫が必要となる．それらの技術や利用法については，別の機会にゆずることとしよう．

〔図29〕ゆらぎとその吸収

〔図30〕H.323とSIPの呼制御

(a) H.323の場合　(b) SIPの場合

参考文献

1）Charles E. Spurgeon著，桜井豊 監訳，柏木由美子 訳，『詳説イーサネット』，オライリージャパン，2000年，ISBN4873110262
2）James D. Carlson著，榊原一矢 監訳，『PPP - 設計・実装・デバッグ -』，オーム社，2002年，ISBN4274064778
3）John M. Davidson著，後藤滋樹，村上健一郎，野島久雄 訳，『はやわかりTCP/IP』，共立出版，1991年，ISBN4-320-02562-8
4）Ravi Malhotra 著，清水奨 訳，『入門IPルーティング』，オライリージャパン，2002年，ISBN4-87311-085-8
5）Van Jacobson, Michael J. Karels, "*Congestion Avoidance and Control*", *ACM Computer Communication Review*; Vol.18, no.4, pp.314-329, August, 1988
6）V. Jacobson, "*Berkeley TCP Evolution from 4.3-Tahoe to 4.3-Reno*", *Proceedings of the Eighteenth Internet Engineering Task Force*, p.365, 1990.
7）W.Richard Stevens著，橘康雄 訳，井上尚司 監訳，『詳細TCP/IP』Vol.1 プロトコル，ピアソン・エデュケーション，2001年，ISBN4-89471-320-9
8）飯塚久夫 監修，大西廣一，岡田忠信，和泉俊勝，大宮知己著，『IP時代のやさしい信号方式』，電気通信協会，2002年，ISBN4-88549-414-1
9）Bill Douskalis, *IP Telephony*, Prentice Hall，2000，ISBN 0-13-014118-6

むらかみ・けんいちろう　NTT研究所

New Products ― コンテック，完全フロントメンテナンスタイプのウォールマウント型組み込みパソコンを発売
（株）コンテックは，メンテナンス性/省スペース性を追求した，ウォールマウント型の組み込みパソコン用ATXシャーシ「MPCシリーズ」を発売した．ユニットの引き出し機構を採用し，完全にフロント面から配線や部品交換などのメンテナンスができるのが特徴．オープン価格．

VoIPの基本概念と用語を理解する
VoIP技術の基礎知識

和泉俊勝

VoIPとは，IPネットワーク上での通話を可能にする技術である．VoIPによる通話は，相手側の端末を検索/接続し，使用する音声形式などの必要な情報をやりとりし，実際にディジタル化された音声データを相手と送受信するという流れになる．

本章では，これらのVoIPの通信における一連の流れと，VoIPに特有な用語の解説などを行う．
（編集部）

はじめに

DSL，光アクセスなどによるアクセス回線のブロードバンド化および，常時接続利用形態の進展を背景にして，インターネットの広範な利用が始まっている．とくに，VoIP（VoIPとは「Voice over IP」の略で，電話をIPネットワーク上で使えるようにするための基盤技術）技術の進展によって，インターネット接続事業者によるIP電話サービスが次々と始まっている．

Yahoo BBのIP電話サービスが，1年足らずで160万ユーザーを獲得し，それに刺激されたかのように既存の大手電話キャリアもIP電話に参入し始めている．さらにこれから，電話網との相互接続が本格化する様相を見せており，これまでの電話網を中心とした電話から，IPネットワークによる電話サービスへと急激なシフトが予想されている．

本章では，このような電話サービスの電話網からIPネットワークへのシフトをふまえ，IPネットワーク上で電話を実現するための技術であるVoIPについて解説する．

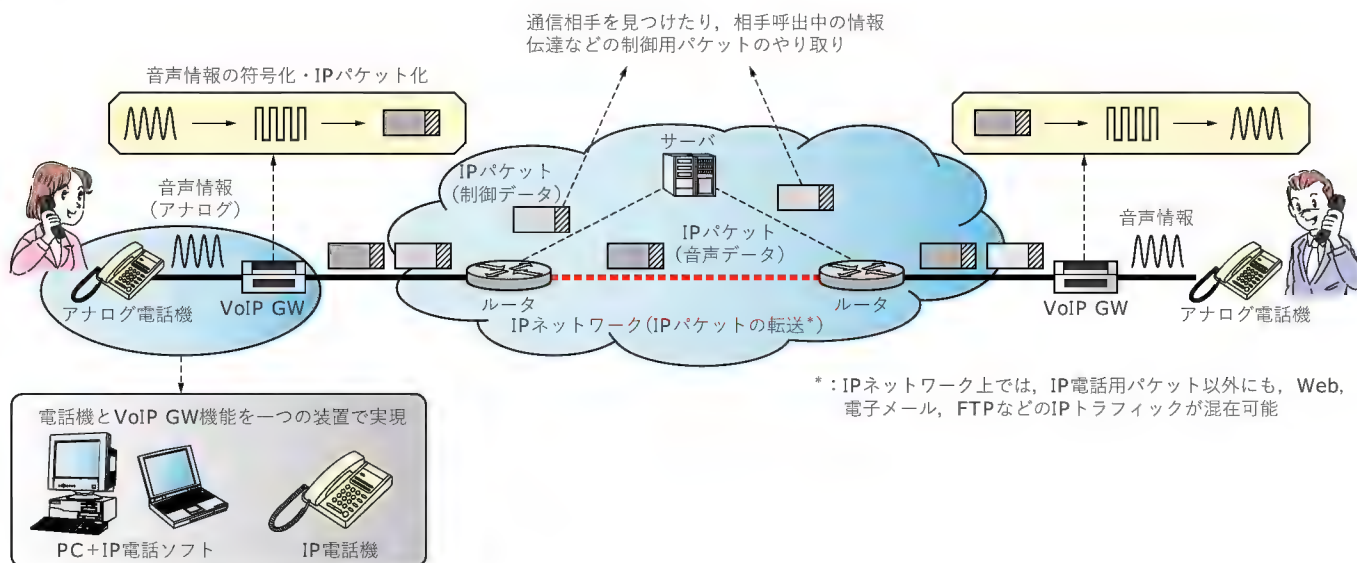
最初に，VoIPの基礎を理解するため，音声をディジタル化，パケット化し，IPネットワークで音声情報を転送し，通話を可能にするしくみを簡単に説明し，それから，VoIPの基礎となる，i）音声パケット通信技術と通信品質，ii）IP電話の発信，着信のための呼制御プロトコル，iii）電話網との相互接続技術，について述べることとする．

なお，ここでは，IPネットワーク上の電話を「IP電話」と呼び，従来の電話網の電話を「固定電話」と呼ぶことにする．

IP電話のしくみ

IP電話のしくみを**図1**に示す．

1）IP電話を実現するためには，まず，音声（アナログ）情報をIPパケットにしなければならない．音声をディジタル情報に変換し，それをIPパケット化する．これは，ユーザー側の「VoIPゲートウェイ（VoIP GW）」と呼ばれる装置で実現される．VoIP GWでは，音声情報のIPパケット化のほかに，音

〔図1〕IP電話のしくみ

声情報の送り出し先（通信相手）を見つけたり，相手呼び出し中の情報伝達など，IP電話に関する制御用の通信も行う

2）VoIP GWでIPパケット化された音声情報は，IPネットワーク上で，ほかのIPパケットと同様に，宛先IPアドレスを見て，転送される

3）IPパケットが相手側VoIP GWまで届いたら，1）と逆の処理で，IPパケット化された音声情報が音声（アナログ）化され，届けられる

2 音声パケット通信技術と通信品質

2.1 IP電話の品質クラス

IP電話の品質クラスとして，わが国では総務省の「IPネットワーク技術に関する研究会」で，ITU-T（International Telecommunication Union-Telecommunication Standardization Sector），ETSI（European Telecommunication Standards Institute）のTIPHN（Telecommunications and Internet Protocol Harmonization Over Networks）および米国のTIA（Telecommnications Industry Asssociation）におけるIP電話の品質クラスとの整合を図りながら，**表1**の3クラスに分類している．

これは，IP電話のサービスが，ベストエフォートのインターネットを利用するサービスから従来の固定電話と同等のサービスまで，さまざまなグレードでのサービス提供を想定している．また，IP電話として，クラスCに満たないものは，ユーザーが電話サービスとして利用するには困難な品質と考えられると述べている．

2.2 音声パケット通信技術と通信品質への影響

IP電話では，音声情報をディジタル符号化したのちにIPパケット化するが，そのIPパケットをIPネットワーク上のルータでは，いったん蓄積してから転送するため，遅延とジッタ（遅延揺らぎ），パケットロスが発生する．このため，受信したIP電話パケットをそのままの状態で音声に復元しても，音声が途切れたりして，スムーズに再生できないことがある．

〔表1〕IP電話の品質クラス分類

	クラスA（固定電話並 注）	クラスB（携帯電話並 注）	クラスC
総合音声伝送品質率（R）	> 80	> 70	> 50
End-to-End遅延	< 100ms	< 150ms	< 400ms
呼損率（接続品質，参考値）	≦ 0.15	≦ 0.15	≦ 0.15

● R値，遅延に関する表中の数値は，95%確率で満足させるものとする．
注：ここでの固定電話並，携帯電話並とは，それぞれ通信品質のうち総合音声伝送品質（R）に注目した場合を表し，End-to-End遅延やその他の機能などについて既存の固定電話並または携帯電話並を求めるものではない．
［IPネットワーク技術に関する研究会報告（総務省2001.6-12）より］

1）遅延

遅延は，スムーズな会話のためには極力小さく抑える必要がある．IP電話のクラス分類からもわかるように，片道400msを超えると通話に大きな支障が出るといわれている．おもな遅延の要素は，送信側の符号化時間，パケット化時間，受信側では，ジッタ吸収バッファ時間，復号化時間，そして，これにネットワーク遅延時間が加わることになる．

遅延の削減には，遅延の少ないCODECを選定すること（CODECの詳細については，特集第4章を参照のこと）．また，パケット化間隔を小さくすることなどが有効である．

2）ジッタ（遅延揺らぎ）

送信側で一定間隔で送出したパケットが，受信側では等間隔には到着しない．音声パケットの到着間隔が長くなると，通話中に音声がブツブツと途切れるなどの状態が発生する．ジッタを吸収するためには，

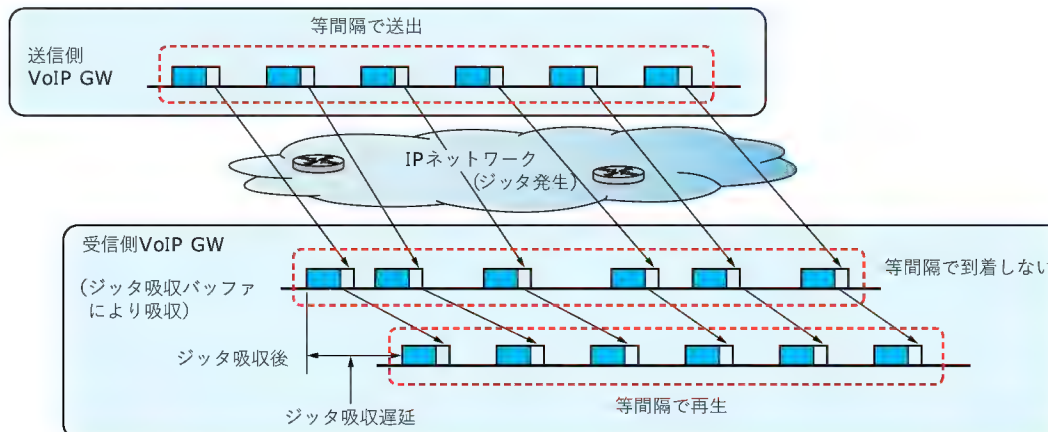
i）**図2**のように，受信側でジッタ吸収バッファを設けて吸収する
ii）ネットワーク内でジッタ発生を抑えるような工夫をする（後述）

ことが必要となる．

3）パケットロス

パケットロス（パケットの欠落）があれば，当然受信側では正しく音声を再生できない．受信側でパケット欠落を検出して送信側から再送する方式は遅延が増大し，IP電話には適さない．

〔図2〕受信側でのジッタの吸収

New Products ─ スペクトラテック，12.8ビット（7141階調）を達成したモノクロLCDパネルを発売
（株）スペクトラテックは，12.8ビット（7141階調）を表現可能とする技術を確立し，20.8インチモノクロQXGA（2048画素×1536ライン）LCDパネルを開発した．この技術は，X線フィルムからディジタル化して保存/診断しようとする医療分野のIT化に対応するものである．

そのためIP電話では，受信側で欠落したIP電話パケットを補完する方式と，送信側であらかじめIP電話パケットに冗長な音声情報を付与し，冗長な音声情報から欠落した電話IPパケットを復元する方式がある．

2.3 ネットワークのQoS保証技術

通信品質の劣化の要因となる遅延，ジッタ，パケットロスに対するネットワークのQoS保証技術について，もう少し解説する．IPネットワークで，これらの劣化要因が増加する背景として，IP電話パケット以外のさまざまな長さのIPパケットが同じIPネットワークに流れ込んでくることがある．その量と処理の関係で，IP電話パケットの遅延，ジッタ，パケットロスに影響が出てくる．

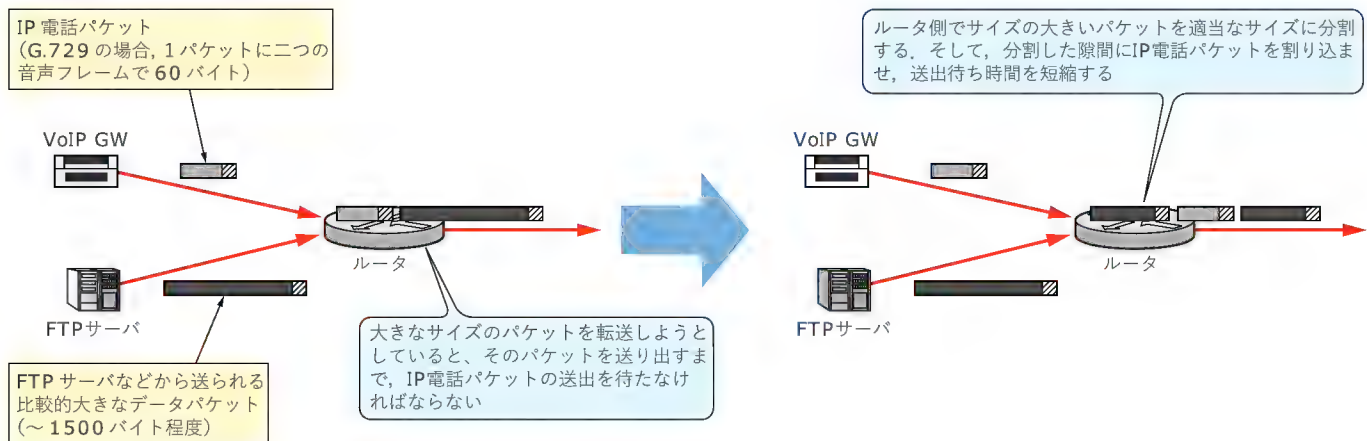
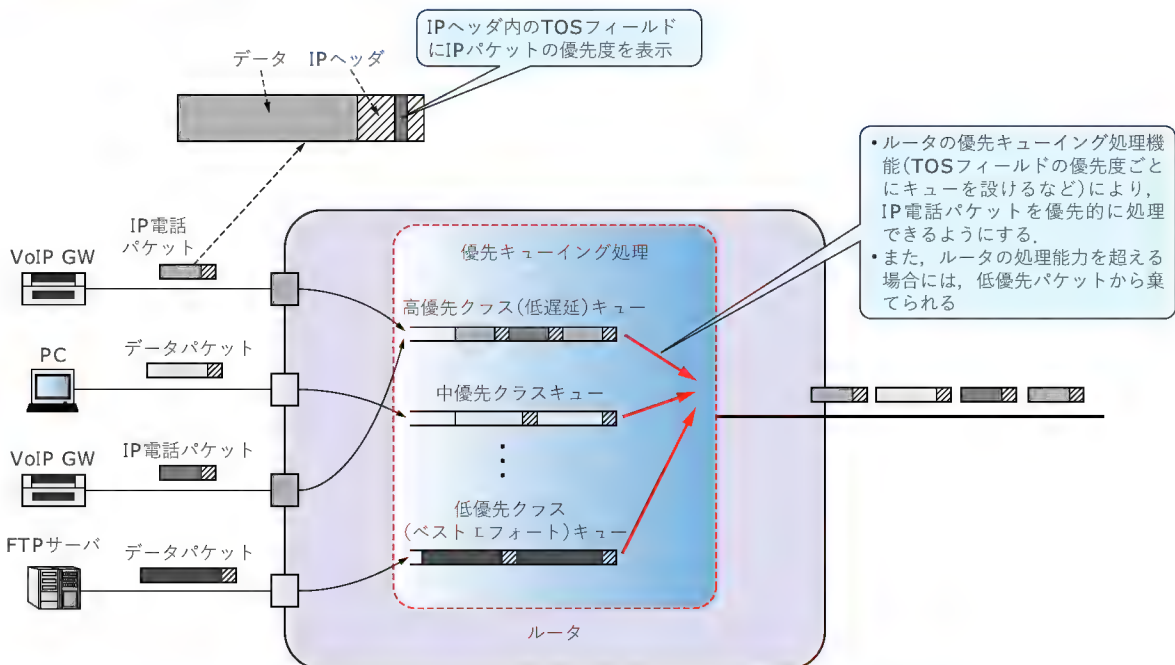
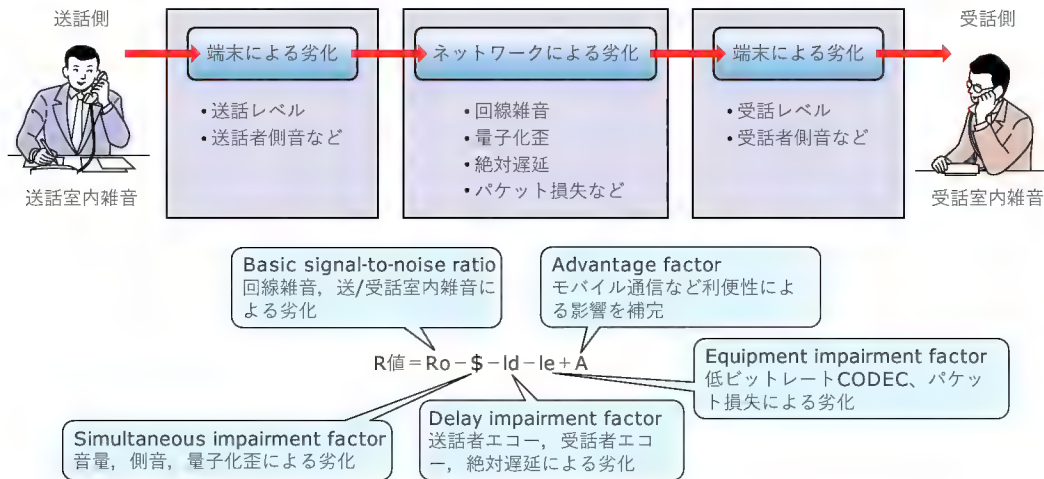
たとえば，IP電話パケットが到着したときに，ルータがIP電話パケット以外の大きなサイズのパケットを転送しようとしているとする．結果として，そのパケットを送り出すまでIP電話パケットの送出が待たされ，遅延の原因となる．この遅延は，とくに低速回線で大きな問題となる．

また，ルータはIPパケットを受信すると，いったんキューと呼ぶ待ち行列に並べ，これを先着順に処理する．IPパケットが増大し，ルータのキューが満杯になると，キューから溢れたパケットは廃棄される．IP電話以外のIPパケットの増大により，IP電話パケットが廃棄されたり，処理が遅れたりするような事象が起こる．

このような事象への対策として，**図3**に示すように，ルータ側でサイズの大きいパケットを適当なサイズに分割するフラグメンテーション機能を利用し，分割した隙間にIP電話パケット

〔図3〕フラグメンテーション機能

〔図4〕優先キューイング処理機能

New Products ― 日立，都市間通信ネットワーク向けMEMS型光スイッチ素子を発売
(株)日立製作所機械研究所は，日立金属(株)先端エレクトロニクス研究所の協力を得て，都市間(メトロ)ネットワーク向けの3次元のMEMS(Micro Electro Mechanical Systems)型光スイッチ素子を開発した．この素子は，3次元のMEMS型光マトリクススイッチ方式を採用している．

〔図5〕**E-modelの概要**(R値を構成する要素)

を割り込ませて送出待ち時間を短縮するとか，**図4**に示すようにルータの優先キューイング処理機能により，IP電話パケットを優先的に処理できるようにするなどの工夫が必要となる．

優先キューイング処理では，IPパケットのヘッダ部分にあるTOS(Type Of Service)フィールドを利用し，これをルータが識別し，特定のIPパケットを先に処理する(DiffServeとしてIETF：Internet Engineering Task Forceで標準化)．このほか，あらかじめIP電話パケットのために使用する帯域の予約を図る方式(RSVP：Resource Reservation ProtocolとしてIETFで標準化)などがある．IP電話の品質クラスA，Bなどは，このような工夫によるQoSの保証が可能なIPネットワークでの実現が必要となる．

2.4 R値

IP電話の品質クラスで記述している「R値」について，参考として示す．

音声通話品質に関して，ITU-Tでは，一般的な主観評価方法として，被験者に音声の品質を5段階(1：bad～5：excellent)で評価してもらい，その平均値で品質を表すMOS(Mean Opinion Score，ITU-T勧告P.800)がある．

また，客観評価については，音声通話の品質を設計するモデルとして**図5**のE-modelを構築し，その上で総合音声伝送品質を表すR値を定義している(ITU-T勧告G.107)．また，ITU-T勧告G.109で，R値による総合音声伝送品質のカテゴリ分類を定めている(**表2**)．

〔表2〕**R値による総合音声伝送品質のカテゴリの定義**

R値の範囲	カテゴリ	ユーザー満足度
100 ≧ R ＞ 90	Best	Very satisfied
90 ≧ R ＞ 80	High	Satisfied
80 ≧ R ＞ 70	Medium	Some users dissatisfied
70 ≧ R ＞ 60	Low	Many users dissatisfied
60 ≧ R ＞ 50	Poor	Nearly all users dissatisfied

3 呼制御(セッション制御)プロトコル

IP電話は，従来の電話と同様に相手と通話するためには，電話番号をダイヤルするとか，IP電話ソフトの電話帳から接続PC名称などを指定して相手を呼び出さなければならない．相手側では，着信があったときに電話機のベルを鳴らすとか，通信を行う場合は応答を返すなどをしなければならない．このように，IP電話の呼(call)を管理するためには呼制御(セッション制御)機能が必要となる．

IP電話では，この呼制御機能を，音声情報との通信とは別に，制御情報の通信を用いて実現している．そして，その通信の実現に，呼制御(セッション制御)プロトコルを使う．呼制御プロトコルとして，VoIP GW装置やIP電話ソフトで一般ユーザー向けとしてよく採用されているのが，H.323とSIPである．

3.1 H.323

H.323の詳細については，第3章や第7章で詳細に記述しているので，ここでは，基本的な説明だけをする．

H.323に記述されている通信システムの構成，プロトコル群を**図6**に示す．H.323は，テレビ会議などのマルチメディア通信向けのプロトコル群としてITU-Tが標準化したもので，一部の機能を使って，IP電話を実現している．古くから使われており，実績も多い．

1）H.323の構成要素

① H.323端末

IP電話通信を行う端末である．音声・動画像の入出力機能，H.323接続制御機能を有する．PC上のソフトあるいは拡張ハードウェア，あるいは，専用のIP電話機などがある．

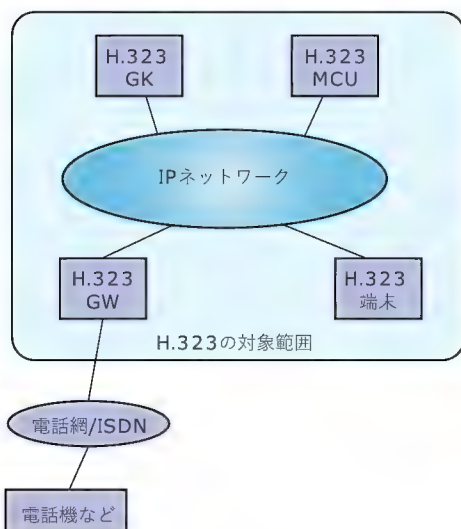
② H.323ゲートウェイ(H.323GW)

VoIP GWに相当し，既存の電話網や，PBX，または，電話端末などと接続し，呼制御信号や音声情報などの変換処理を行う．

④ H.323ゲートキーパ(H.323GK)

ゾーン(Zone)と呼ばれるIPネットワーク上の管理領域に存

New Products ── ギガビット対応PCIバス用LANアダプタ
(株)コレガは，32ビットPCIバス接続のギガビットEthernet対応LANアダプタ「corega GEther PCI-T32」を発売した．最速の1000Base-T接続をはじめ，従来の100Base-TX/10Base-T通信も，オートネゴシエーション機能搭載により自動認識することができる．

〔図6〕H.323通信システムの構成とプロトコル群

(a) H.323通信システムの構成

音声コーデック部		ビデオコーデック部		システム制御部		
音声圧縮データ	音声フロー制御	映像圧縮データ	映像フロー制御	ネゴシエーション	呼の確立/解放	アドレス解決 受付可否制御 帯域制御
G.7xx (G.711, G.729, ...)	RTCP	H.26x (H261, H.263)	RTCP	H.245制御	H.225.0 呼制御	H.225.0 RAS制御
RTP		RTP		TPKT	TPKT	
UDP	UDP	UDP	UDP	TCP	TCP	UDP
IP						
リンクレイヤ						
物理レイヤ						

TPKT：TPDU(Transport Protocol Data Unit)Packet
メッセージにメッセージ長を示すヘッダを付加する規格

(b) H.323通信システムのプロトコル群

在するH.323端末，H.323GWなどに対して，H.225.0という呼制御プロトコルで呼制御を提供する．

2）H.323のプロトコル群

① H.225.0RAS制御およびH.225.0呼制御

H.225.0RAS制御は，H.323端末，H323.GWなどのH.323エンドポイントとH.323GKとの間のプロトコルで，通信開始に先立って端末の登録，相手先アドレスの解決[注1]などに使う．一般に，UDPで転送する．

H.323.0呼制御は，H.323エンドポイントとの呼設定，解放などの処理を行う．これは，ISDNの呼制御メッセージをベースとしている．この手順は，H.323エンドポイント間で直接通信する場合とH.323GKを介して通信する場合がある．一般にTPKT/TCPで転送する．

② H.245制御

呼確立後，H.323エンドポイント間の論理チャネルの設定，解除を行う．ネゴシエーションにより，使用する符号化などを決めることができる．一般にTPKT/TCPで転送する．

③ RTPおよびRTCP(H.225.0)

RTPは，音声や動画像などのリアルタイム情報を転送するためのタイムスタンプを含むパケット形式を規定している．またRTCPは，RTPで転送される情報に対するフロー制御や基準時間情報を伝達する手段を提供する．

3）H.323の具体的な動作概要

図7に，H.323の具体的な動作概要を示す．

① 端末の登録

H.323端末とH.323端末間で通信を行う場合に，最初にH.323GKにH.323端末の登録を行い，H.323端末が通信可能な状態とする．

i) H.323端末からGKにGRQ(GateKeeper Request)メッセージの送出する

ii) GRQメッセージを受信したH.323GKは，H.323端末に対して，GCF(Gatekeeper Confirm)メッセージを返送する．その後のH.225.0RAS制御で用いるH.323GKのRAS制御用アドレスを通知する

iii) GCFメッセージを受信したH.323端末は，H.323GKに，別名アドレス，呼制御用アドレス，RAS制御用アドレス，有効時間などの情報を含むRRQ(Registration Request)メッセージを送信する

iv) RRQメッセージを受信したH.323GKは，登録を行い，RCF(Registration Confirm)メッセージをH.323端末に返送する．この手続きにより，H.323GKは，別名アドレスによる接続要求に対して，呼制御用アドレスの検索が可能となる

② 呼の設定

次に，エンドtoエンドの通信を行うためのアドレス解決と呼の設定を行う．

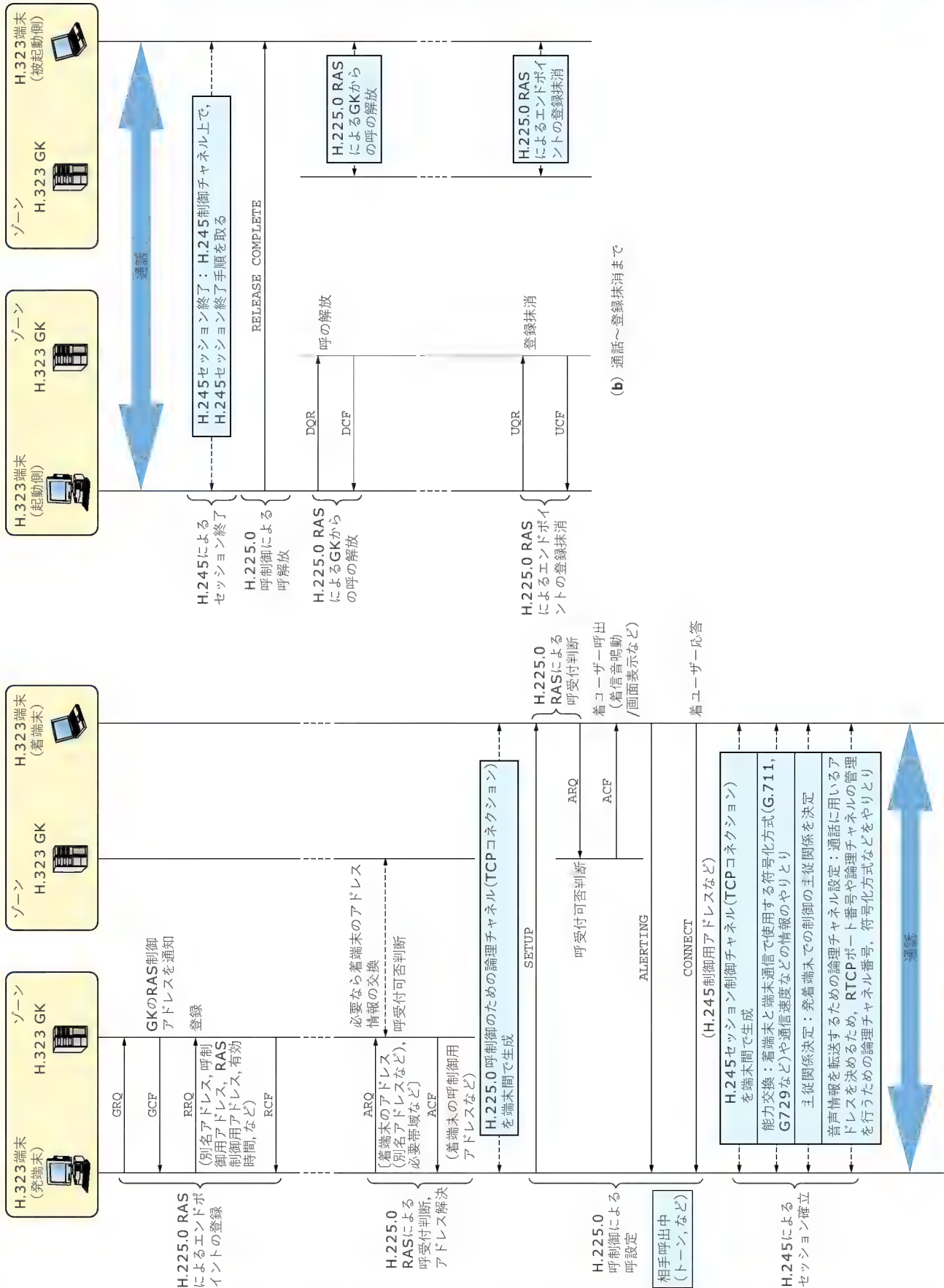
i) 発H.323端末は，H.323GKにARQ(Admission Request)メッセージを送出する．このメッセージには，着端末のアドレス(別名アドレスなど)や必要帯域などを含む

ii) ARQメッセージを受信したH.323GKは，呼の受け付けを許可する場合，ACF(Admission Confirm)メッセージを着端末に返送する．このメッセージには，着端末の呼制御用アドレスを含む．着端末がほかのH.323GKに属していて，発側GK

注1：IPネットワークにおける端末の識別を行うIPアドレスは，各IPネットワークの管理者によって各端末に割り当てられる．また，同一端末での多種な通信を識別するためにポート番号を使用する．このため，IP電話で通信を行うためには，通信相手のIPアドレスやポート番号を取得する必要がある．H.323では，電話番号やホスト名から，これらの情報を取得する手順(これを「アドレス解決」と呼ぶ)を提供する．

New Products ― 音響・振動計測で5,000チャネルの同期を可能にする同期モジュール
日本ナショナルインスツルメンツ(株)は，音響，振動，ノイズ，動的圧力の計測データについて最大5,000チャネルを同期させ，集録，解析，表示するシステムの構築を可能にする新製品「NI PXI-6653 同期モジュール」を発売した．価格は￥421,000．

〔図7〕H.323の動作概要

New Products ― 中抜きの薄型ロータリエンコーダスイッチ
ミツミ電機(株)は, ほかのスイッチを搭載可能な中抜きの薄型ロータリエンコーダスイッチ「SREシリーズ」を発売した. 携帯電話やPDA, デジカメなどのポータブル機器で, 複合操作スイッチが求められており, また同時に小型化も要求されているためである.

に情報がない場合，発側GKは，情報をもつ他H.323GKとの間でアドレス情報交換などの処理を行う

iii）AFCを受信した発端末は，着端末の呼制御用アドレスに対してTCP/IPの手順により，H.225.0呼制御のための論理チャネル(TCPコネクション)を端末間で生成する．そして，その論理チャネル上でSETUPメッセージを相手端末に送出する

iv）SETUPメッセージを受信した着端末は，その端末の登録したH.323GKに，通信を開始してよいかの問い合わせのため，ARQメッセージを送信する

v）ARQメッセージを受信したH.323GKは呼を許可する場合，着端末にACFメッセージを送出する

vi）ACFメッセージを受信した着端末は，着信音を鳴動させたり，画面表示させるなど，着端末ユーザーに着信を通知し，ALERTINGメッセージを発端末に返送する

vii）ALERTINGメッセージを受信した発端末は，発端末ユーザーに相手呼出中であることをトーンなどで通知する

viii）着端末ユーザーのIP電話への応答の操作により，着端末は，CONNECTメッセージを発端末に送信する．このメッセージには，この呼で使用するH.245制御用アドレスを含む

ix）発端末がCONNECTメッセージを受信し，呼が確立する

③ セッションの確立

呼確立の後，発端末と着端末の間で通話のためのネゴシエーションを行いセッションを確立する．

i）発端末は，CONNECTメッセージで取得したH.224制御用アドレスに対して，TCP/IP手順によりH.245制御チャネル(TCPコネクション)を生成する

ii）そして発端末は，H.245制御用チャネルを介して着端末と端末通信で使用する符号化方式(G.711，G.729など)や通信速度などの情報のやりとりを行い，お互いの端末が利用可能な情報などを認識する

iii）次に，発着端末での制御の主従関係を決定する．この主従関係は，マルチポイント会議における接続制御などでの競合が起きたときの優先順位を定めるものである

iv）続いて，音声情報を転送するための論理チャネルを設定する．通話に用いるアドレスを決めるため，RTCPポート番号や論理チャネルの管理を行うための論理チャネル番号，符号化方式などをやりとりする．これは，発端末から着端末方向に音声情報を送信するための論理チャネルの設定，着端末から発端末方向への論理チャネルの設定，さらには，テレビ会議など，複数メディアを用いての通信の場合には，各々の論理チャネルを生成する

④ 通話

音声通話用の論理チャネル生成時にネゴシエーションされた情報にしたがって音声情報を符号化し，RTPで音声情報を送受信し，音声通話を行う．

⑤ セッションの終了と呼の解放

呼の解放は，H.323端末からの呼の解放の場合と，H.323GKからの呼の解放の場合があるが，ここでは，端末からの呼の解放を示す．また，解放を起動する端末を起動側端末，もう一方を被起動側端末と呼ぶ．

i）まず，起動端末，被起動端末間で，H.245セッションを終了するため，H.245制御チャネル上で，H.245セッション終了手順を取る

ii）続いて，H.225.0呼制御チャネルが残っている場合は，RELEASE COMPLETEメッセージを送出する

iii）各端末は，自分の帰属するH.323GKにDRQ(Disengage Request)メッセージを送信する

iv）DRQメッセージを受信したH.323GKは，DCF(Disengage Confirm)メッセージを返送し，呼を解放する

⑥ 端末の登録取り消し

H.323端末からH.323GKに登録を取り消す場合，

i）H.323端末からH.323GKにURQ(Unregistration Request)メッセージを送信する

ii）URQメッセージを受信したH.323GKは，登録を抹消し，UCF(Unregistration Confirm)メッセージを返送する

このほか，H.323通信では，H.323GKを介して呼制御メッセージの交換を行うゲートキーパ経由呼シグナリングが可能であったり，ここで，説明した以外のいくつかの基本的な手順(位置情報要求，状態要求など)，付加サービスを提供するためのH.450拡張呼制御手順などがある．

また，H.323通信におけるセキュリティとして，途中経路での盗聴やなりすまし通信など不正を防止するため，ITU-T勧告H.235で認証や暗号化などの手段を規定している．

H.323通信は，これまで説明したきたように，マルチメディア通信をターゲットとしているため，IP電話の通信に限っての利用においては，すべての仕様に準拠する必要がない．そこで，IP電話専用に用いることを想定し，H.323Annex F(SET：Simple Endpoint Type)が規定された．これは，H.225.0呼制御メッセージの中にH.245制御メッセージの論理チャネル生成を含めるファーストコネクト手順などにより，接続時間を短縮するなどを実現する．

3.2 SIP

SIPの詳細については，第3章で記述しているので，基本的な説明だけにする．

SIPは，インターネットに代表されるIPネットワーク上での音声や，映像などのストリーム型の通信のセッションの設定や解放などの各種制御機能を提供するプロトコルとして，IETFが標準化した．SIPは，インターネット仕様をベースとしており，呼制御のやり取りをWebアクセスと同様のリクエスト-レスポンス型で行うなど，既存のインターネット技術を活用している．またSIP自体は，セッション制御に利用するメッセージのフォーマットや交換手順を規定しているだけで，音声や画像メディアの転送方法や，通信システム全体に関しては規定していない．

New Products ── パルテックの並列処理用コンピュータシステム
パルテック(株)は，高速化の要求に柔軟な拡張性で対応できる並列処理用コンピュータシステム「MC6」を開発した．このマルチコンピュータシステムは，ボード単位，シャーシ単位，ラック単位で64000CPUまで拡張可能である．

〔図8〕SIPでのネットワーク構成

注：ロケーションサーバとSIPサーバ，レジストラ間は実装依存であり，SIPの規定範囲外である．

〔図9〕SIPプロトコルスタック

RTP/RTCP リアルタイム 情報転送	SIP (Session Initiation Protocol) 呼・セッション制御 ／ SDP (Session Description Protocol) 制御情報記述
UDP	UDP(TCP, SCTP)
IP	
リンクレイヤ	
物理レイヤ	

参照元：RFC2543

SIPは，HTTPとの類似性およびテキスト形式の記述が大きな特徴であること，また，ネットワーク装置に呼状態を保持しないステートレス動作を可能にしているなど，端末機能を重視したネットワーク構成を可能にしている．

1）SIPでのネットワーク構成

図8にSIP(Session Initiation Protocol)でのネットワーク構成，**図9**にプロトコルスタックを示す．

① ユーザーエージェント

端末相当．

② SIPサーバ(プロキシサーバ)

SIPメッセージを処理し，端末の代理として，セッション制御するサーバ．

③ ロケーションサーバ

着端末のユーザー名から，現時点の所在地を導出するデータベース．ロケーションサーバは，ユーザーのURI(Uniform Resource Identifier)から，対応するプロキシサーバやユーザーエージェントのURIを導出する装置であり，複数候補が登録されている場合の変換ルールは運用者が決定する．

④ リダイレクトサーバ

現時点での相手所在地に応じて，SIPメッセージを差し戻すサーバ．

⑤ レジストラ

ユーザーからのREGISTERメッセージを受信し，ユーザーの所在情報をロケーションサーバに登録するサーバ．

2）SIPの具体的な動作概要

図10(次頁)に，SIPの具体的な動作概要について説明する．

① 登録

ユーザーは，現時点での位置情報や宛先を帰属するドメインを管理するレジストラに登録することを可能にする．

i）ユーザーは，レジストラにREGISTERリクエストメッセージを送信する

ii）REGISTERリクエストメッセージを受信したレジストラは，ユーザー名であるSIP URI(Address-of-Record：AORと記述され，ユーザーのパブリックアドレスとみなされる)と，現在使用しているSIPフォンあるいは，パソコンのSIP URIを対応付け(バインディング)，ロケーションサーバのデータベースに登録する．この登録手順を利用することにより，一般に公開するSIP URIとは異なる場所へのSIPメッセージの転送，複数宛先を登録することによる複数端末への同時着信などを可能にする

② 呼の設定

発ユーザーは通信相手の識別子であるSIP URIを用いて着ユーザーを指定し，セッションの設定をする．SIP URIは，E-mailアドレスと似た形式であり，通常，ユーザー名とホスト名を含む．たとえば，sip:izumi@kita.comである．kita.comは，izumiのSIPサービスプロバイダのドメインである．**図11**にSIP URIの形式を示す．

i）発ユーザーは，着ユーザーの端末やドメインのプロキシサーバを知らないので，自ドメインのプロキシサーバに，まず，INVITEリクエストを送出する．INVITEリクエストには発

〔図11〕SIP URIの形式

sip:user:password@host:port;uri-parameters?headers

user：宛先となるhost(あるいは，ドメイン)における特定のリソースを識別する．ホスト自身が特定されるリソースである場合は，省略できる

password：userに関連つけられたパスワード．これの使用については，平文で認証情報を送信するのは危険なため，推奨されていない

host：sipリソースを提供するホスト．Host部分は，FQDN(Fully Qualified Domain Name)，あるいは数値のIPv4またはIPv6アドレスが許されている

port：リクエストが送られるポート番号

uri-parameters：パラメータ名＝パラメータ値の形式をとる．これにより，複数のパラメータを列挙できる．パラメータ例：Transport＝udp(UDPの利用を明示する場合)

headers：ヘッダ名＝ヘッダ値の形式をとる．&により複数のヘッダを列挙できる

New Products — Red Hat Linux 9 発売
レッドハット(株)は，Red Hat Linuxの最新版「Red Hat Linux 9」を発売する．Red Hat Linux 8.0パーソナルおよび同プロフェッショナルの後継製品で，それぞれRed Hat Linux 9と同Professionalの2製品である．価格はRed Hat Linux 9が￥6,800，Professionalが￥19,800．

〔図10〕SIPの動作概要

発SIP端末
発側SIPサーバ（プロキシサーバ）
DNS
レジストラ
ロケーションサーバ
着側SIPサーバ（プロキシサーバ）
着SIP端末

REGISTER
位置情報や宛先のレジストラへの登録
〔SIP URI（Address-of-Record：AOR），現在使用しているSIP端末のSIP URIなど〕
AORと現在使用端末のSIP URIを対応付け（バインディング），ロケーションサーバに登録
登録

INVITE
DNSを検索
INVITE
（経由サーバを示すViaヘッダに発側SIPサーバアドレスが追加）
100 Trying
（受信したINVITEリクエストと同じ，To，From，Call-ID，Cseqヘッダを持つ）
問い合せ
着端末のIPアドレスを通知
応答
INVITE
（経由サーバを示すViaヘッダに着側SIPサーバアドレスが追加）
100 Trying
着ユーザー呼出（着信音鳴動/画面表示など）
180 Ringing
（INVITEのViaヘッダを利用して，戻っていく）
180 Ringing
（Viaヘッダから着側SIPサーバアドレスを削除）
180 Ringing
（Viaヘッダから発側SIPサーバアドレスを削除）
相手呼出中（トーンなど）
200 OK
着ユーザー応答
200 OK
200 OK
ACK
通話
BYE
200 OK

ユーザーが受信可能なセッションの属性をSDP（Session Description Protocol）で示す．自ドメインのプロキシサーバのIPアドレスはあらかじめ設定されているか，DHCPなどで通知される

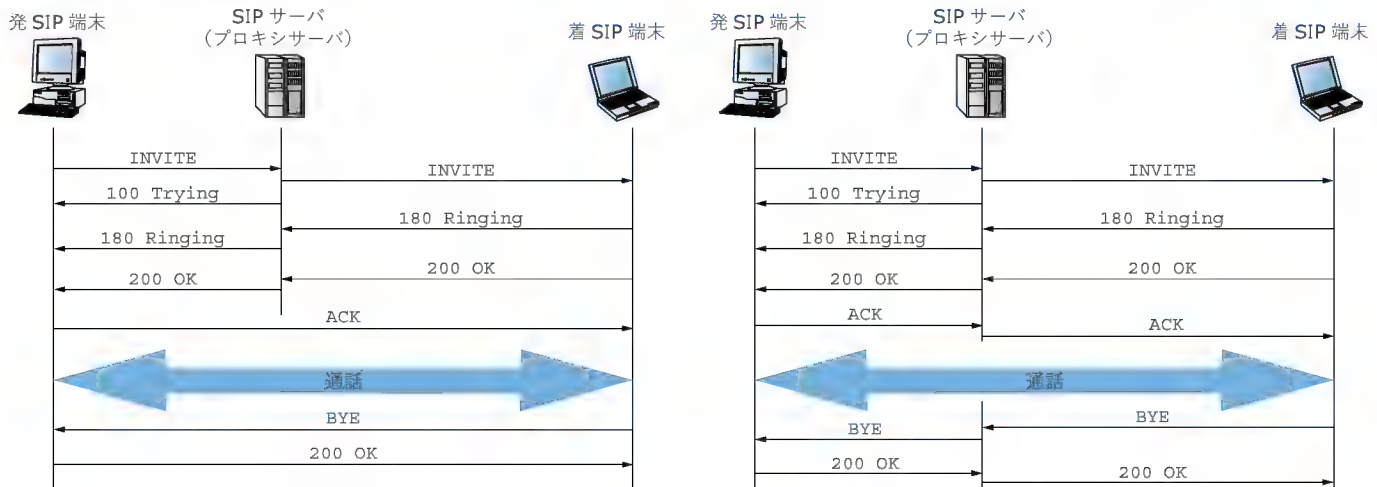
ii）INVITEリクエストを受信した発ユーザーのドメインのプロキシサーバは，発ユーザーの代理として，DNSを検索することにより，着ユーザーのドメインのプロキシサーバのIPアドレスを見つけ，INVITEリクエストをそのIPアドレスに転送する．発側のプロキシサーバは，INVITEリクエストメッセージに，経由サーバを示すViaヘッダに自らのアドレスを追加する．また，発ユーザーに対しては，リクエストに対するレスポンスとして，正常にINVITEリクエストを受信し，相手側に，INVITEをルートすることを試みていることを通知するため，100 Tryingレスポンスを返送する．100 Tryingレスポンスは，受信したINVITEリクエストと同じ，To，From，Call-ID，Cseqを持ち，INVITEリクエストと対応づけられる

iii）INVITEリクエストを受信した着側のプロキシサーバは，着ユーザーの現時点のIPアドレスを管理しているロケーションサーバにアクセスし，得られた着ユーザーのIPアドレスに対してINVITEリクエストメッセージを転送する．このメッセージにも，経由サーバを示すViaヘッダに自らのアドレスを追加する

iv）INVITEリクエストを受信した着ユーザーは，呼出を始め，発ユーザーに向かって180 Ringingレスポンスメッセージを送信．このメッセージは，Viaヘッダを利用することにより，プロキシサーバに戻っていく

v）180 Ringingレスポンスメッセージを受信したプロキシサーバは，レスポンスメッセージをどこに送るかを決定するため，Viaヘッダを使用し，先頭から自アドレスを取り除き，180 Ringingレスポンスメッセージを転送する．結果として，最初のINVITEリクエストメッセージをルートするためにDNSとロケーションサービスのルックアップが必要とされたが，180 Ringingレスポンスメッセージは，ルックア

New Products ― NORTi対応のeBinder
イーソル(株)と(株)ミスポは，イーソルのRTOSベースシステム向け開発スイート「eBinder」をミスポのμITRON4.0準拠「NORTi」に対応させた新製品「eBinder for NORTi」をリリースした．イーソルが販売およびユーザーサポートを一括して担当する．

〔図12〕SIPプロキシの動作モード

(a) ステートレスモード(状態を管理せず，メッセージの転送のみ)

(b) ステートフルモード(セッションの開始と終了を管理)

(c) トランザクションステートフルモード
(INVITEに対し200 OKを受信するまで状態を管理)

ップやプロキシで保持されている状態を使用せずに，発ユーザーに返送することができる

vi) 180 Ringingレスポンスメッセージを受信した発ユーザーは，呼出音もしくは画面表示で，発ユーザーに通知する

vii) 着ユーザーが受話器を取ると，着端末は，応答したことを示すため200 OKレスポンスメッセージを送信する．このメッセージには，着ユーザーが発ユーザーと確立することを望むセッションのタイプのSDP記述をもつメッセージボディを含む．結果として，INVITEリクエストと200 OKレスポンスの2フェーズの交換によって，セッションタイプのネゴシエーション機能を実現している

viii) 発ユーザーが200 OKレスポンスメッセージを受信すると，発ユーザー端末は，200 OKレスポンスの受信を確認するため，ACKメッセージを着信端末に送る．ACKメッセージは二つのプロキシをバイパスし，発端末から着端末に直接送られる．これは，INVITE/200 OKの交換を通じて，発着端末がお互いのアドレスをContactヘッダから知ったことによる．発/着ユーザーのアドレスがお互いに既知となるので，それ以降は，メッセージを直接交換できるようになる．こうして，発/着ユーザー間のSIPセッションは確立し，INVITEトランザクションは完了する

③ 通話

一般に，メディア(音声情報)の転送ルートとSIPメッセージの転送ルートは異なる．また，メディアも双方向で異なるルートを通る．

④ 呼の解放

たとえば，着ユーザーが先に切断するとした場合には，

i) 着端末は，BYEリクエストメッセージを生成する．BYEリクエストはプロキシを経由せず，直接，発端末にルートされる

ii) BYEリクエストを受信した発端末は，200 OKレスポンスメッセージを返送する．これにより，セッションとBYEトランザクションを終了する

この例では，最初のメッセージ交換(呼の設定)までをプロキシサーバ経由としていたが，それ以降は，直接端末間でメッセージ交換している．プロキシサーバにとって，セッションが継続している間，発/着端末間のすべてのメッセージをプロキシサーバ経由とすることで有用な場合もある．その場合，プロキシサーバは，INVITEリクエストメッセージにプロキシサーバのSIP URIを含むRecord-Routeヘッダを追加することにより，すべてのメッセージをプロキシサーバ経由とすることが可能となる．

3) SIPプロキシの動作モード

SIPプロキシには，**図12**に示す三つの動作モードがある．

① ステートレスモード

New Products — 1.8Mspsの絶縁型ディジタイザ機能を実現する新ドライバソフトウェア
日本ナショナルインスツルメンツ(株)は，NIの提供するPXI用6 1/2桁ディジタルマルチメータ「NI PXI-4070 FlexDMM」のもつ1.8Msps高速絶縁型ディジタイザ機能を最大限利用可能にする新しい無料ソフト「NI-DMM 2.1ドライバソフトウェア」を発表した．

状態をまったく保持しない．SIP メッセージの転送を実行するだけであり，状態にかかわる情報は SIP メッセージ自体に記述される．

② ステートフルモード

呼に関する状態を管理する．呼に関するすべてのメッセージを処理する．

③ トランザクションステートフル

トランザクションに関する状態を管理する．トランザクションに関連する一連のメッセージを処理する．

一般に，ステートレスモードのほうが状態を保持しない分，処理が軽くなる．また，状態を保持しないので，同一のプロキシで一連のメッセージを処理する必要はなく，複数のプロキシサーバによる負荷分散や冗長構成が容易である．ステートフルモードは，セッションを意識し，状態を管理することから，セッションと連動させ，通信の帯域確保などを行うことが可能となる．

トランザクションステートフルモードは，対となるリクエストとレスポンスに対し状態を保持することにより，フォーキング（同時に複数の方向に，INVITE リクエストをフォーク状に分配することにより，たとえば，代表機能などを実現するのに利用されるプロトコル）などを意識するような場合に利用される．

4 電話網との接続

4.1 基本的な接続形態

IP 電話の基本的な接続形態として，図 13 に示すように，これまで見てきたような IP 電話端末～IP ネットワーク～IP 電話端末の接続形態のほかに，IP 電話端末～IP ネットワーク～電話網～電話端末の接続形態（電話網との相互接続）が今後本格化していくと想定される．IP 電話から固定電話を呼び出したり，逆に固定電話から IP 電話を呼び出し，通話ができるようになる．ここでは，IP 電話と既存固定電話が相互に通話できるようにするための技術を見ていくことにする．

1）電話網との接続構成

図 14 に，電話網との接続構成を示す．

① VoIP GW

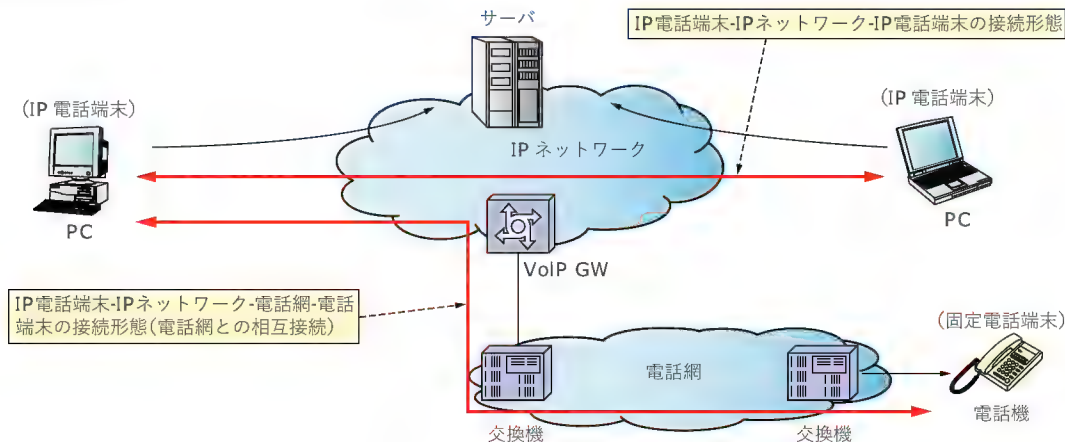
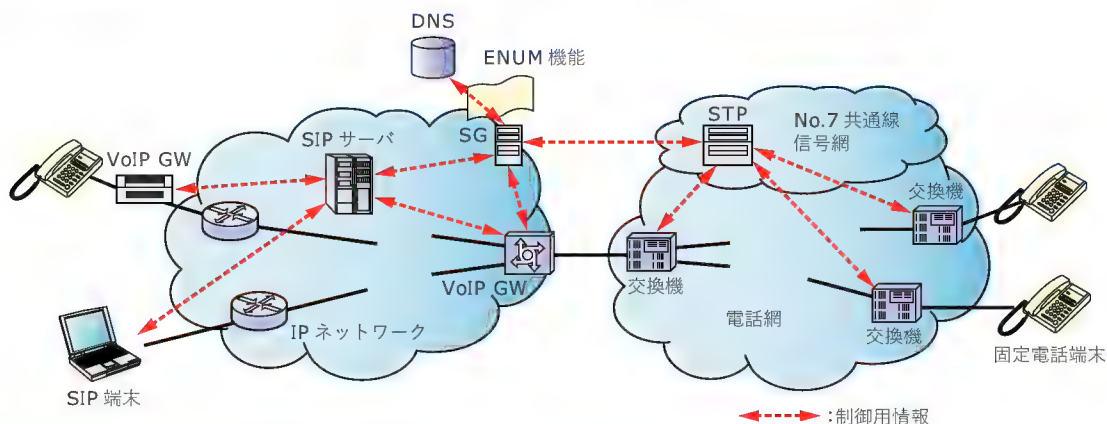
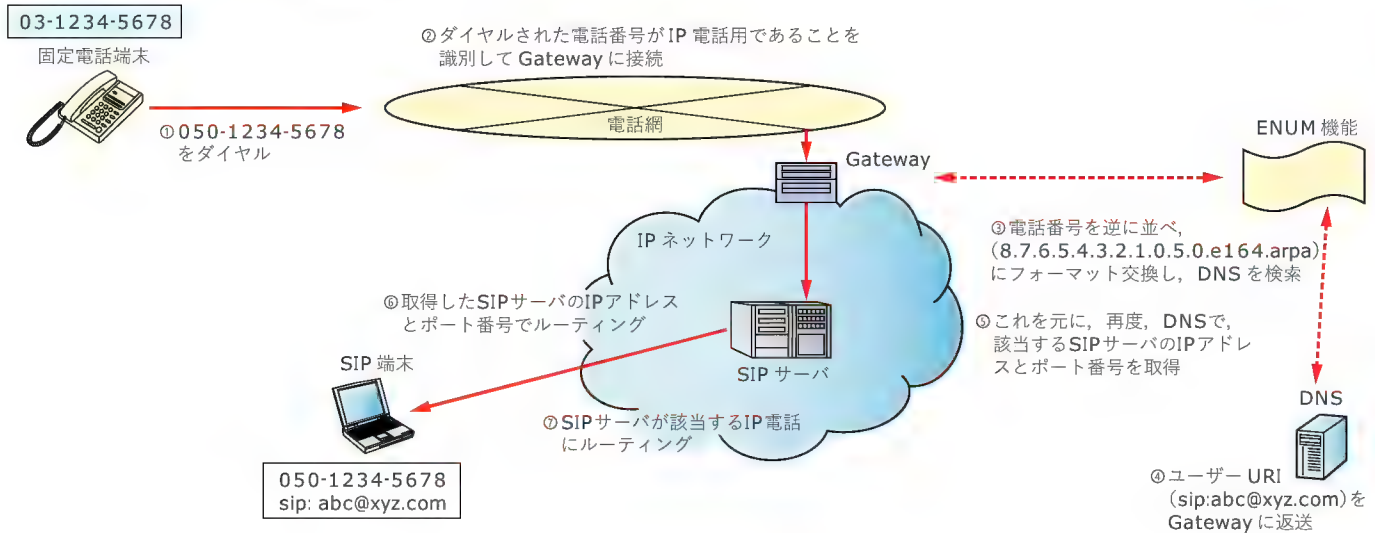
電話網と接続するためのインタフェースを用意した VoIP GW である．

② シグナリングゲートウェイ（SG）

電話網の呼制御プロトコルである No.7 共通線信号方式と接続して制御信号をやり取りするゲートウェイ装置である．

〔図 13〕基本的な接続形態

〔図 14〕電話網との接続構成

New Products ── USB 機器を PC から最大 100m まで延長できるエクステンダ
アイ・ビー・エス・ジャパン（株）（http://www.ibsjapan.com/）は，通常，PC から 5m 程度しか延長できない USB 機器を最大 100m まで延長できる延長機「USB ・ 100 エクステンダー」を発売した．送信装置と受信装置の 2 台から構成され，OS には依存しない．

〔図15〕ENUMにおけるサービス提供例

③ ENUM機能

IP電話に電話番号を与え，電話網からIP電話に電話をかけられるようにするための，電話番号からIPアドレスを対応づけるために使われる機能である．

2）IP電話の番号とENUM機能

IP電話の具体的な番号として，以下の2種類が利用できる．

① ロケーションフリーなサービスへの対応，IPネットワークへのルーティング(振り分け)や事業者識別の容易性，番号容量の確保，ユーザーの利便性の観点から，050番号を利用する．この番号は，一種/二種電気通信事業者の区別なく利用できる

② 第一種電気事業者が，既存の固定電話相当のIP電話サービスを提供する場合は，0AB～J番号を利用できる

ENUMは，電話番号を使って，電話，E-mailなどのIPネットワーク上のさまざまなアプリケーションに接続するために考えられた機能である．具体的な手順について**図15**に示す．

① 通常の電話番号をドメイン名に変換する

② 次に，ドメイン名をDNSサーバに照会し，着信先のURIを得る

③ ユーザーURIから対応するIPアドレスに変換し，着信先に接続する

3）電話網接続の動作概要

IP電話から固定電話に着信する場合をSIPを例に説明する(**図16**)．

① 呼の設定

i) SIP側からINVITEリクエストメッセージをシグナリングゲートウェイが受信すると，シグナリングゲートウェイは電話網の呼制御プロトコルであるISDNユーザー部(ISUP)へのマッピングを行う．SIPのINVITEリクエストに相当するメッセージは，アドレスメッセージ(IAM)である．IAMメッセージの必須パラメータである着番号は，INVITEリクエストの“Request-URI”に基づき設定される．着番号パラメータの番号種別表示は，URIで示す国番号が自国であれば“国内番号”，他国であれば“国際番号”に設定する．また，IAMの発番号オプションパラメータは，INVITEリクエストの“From”から生成する．マッピングされたIAMメッセージは電話網の交換機に送られる

ii) INVITEリクエストからマッピングされたIAMを受信した電話網の着交換機は，着端末に呼を設定するのに必要な情報を受信したことを示すためアドレス完了メッセージ(ACM)を着側に返送する．また，着側のユーザーには，着信を通知するため呼設定(SETUP)メッセージを送出する

iii) ACMを受信したシグナリングゲートウェイは，ACMに含まれる逆方向呼表示パラメータの“着ユーザー状態表示”の値にしたがって，レスポンスメッセージを決定する．この例では，“着ユーザー状態＝表示なし”として考え，183 Session Progressレスポンスにマッピングする

iv) 着交換機は，着ユーザーを呼び出し中であることを示す呼出メッセージ(ALERT)を着ユーザーから受信すると，発側に呼経過メッセージ(CPG)を送出する

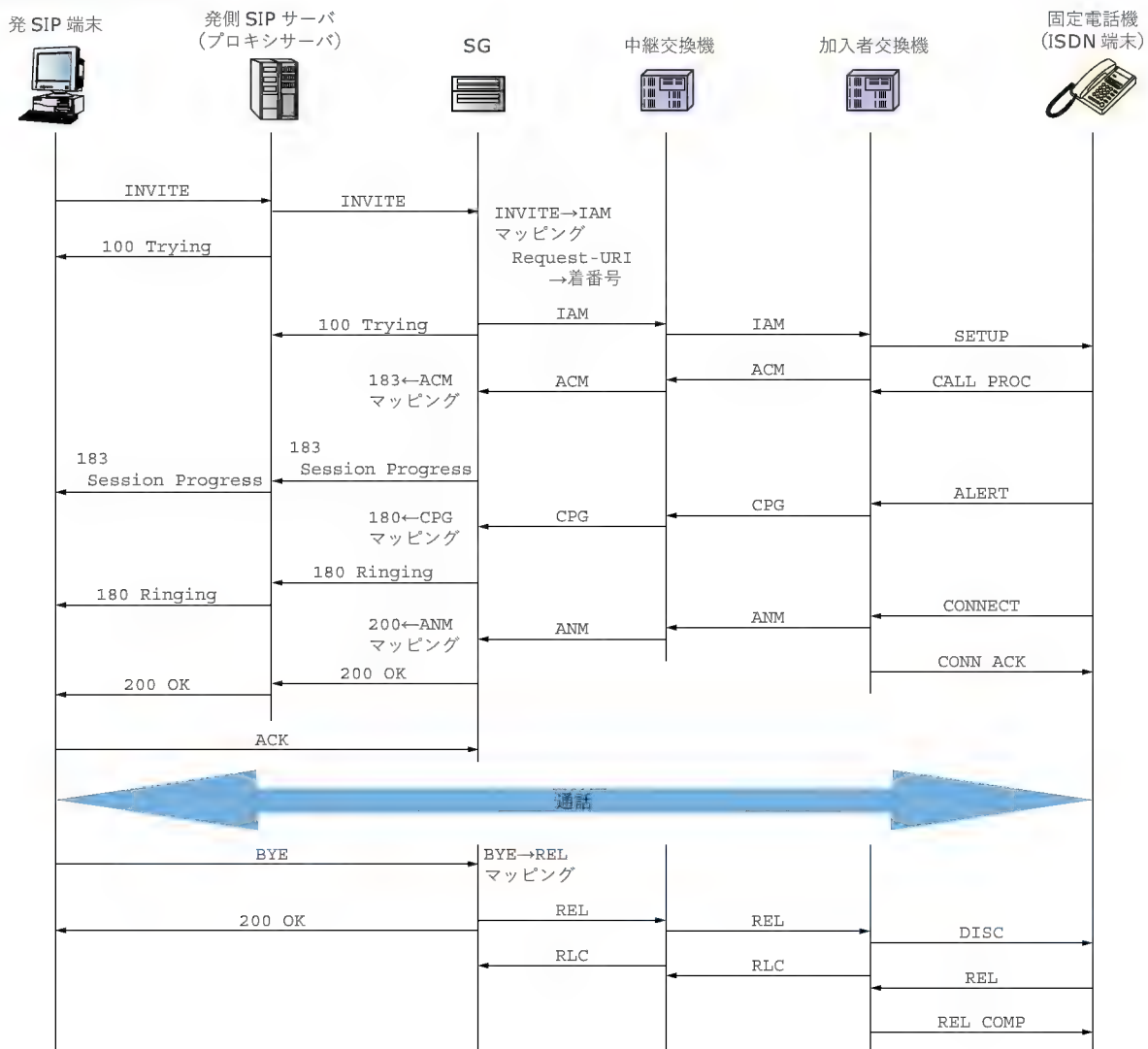
v) CPGを受信したシグナリングゲートウェイは，CPGの含まれるイベント表示パラメータの値に基づき，レスポンスメッセージを決定する．この例では，“イベント表示＝呼出中”なので，180 Ringingレスポンスにマッピングされる

vi) さらに，着交換機は，着ユーザーから応答したことを示す応答メッセージ(CONNECT)を受信すると，発側に応答メッセージ(ANM)を送出する

vii) ANMを受信したシグナリングゲートウェイは，200 OKレスポンスにマッピングし，発側に返送する．発側からのACK受信により発SIP端末とISDN端末の音声通信が可能となる

② 呼の解放

New Products ―ケーブル一体化2：1切換KVMスイッチ
アイ・ビー・エス・ジャパン(株)(http://www.ibsjapan.com/)は，2台のPCに対して一組のキーボード/マウス/モニタからPCをアクセスできるKVMスイッチ「マスタービューACS-62」を発売した．外部電源不要で，VGA最大解像度は1,920×1,440．

〔図16〕電話網接続の動作概要

この例では，発側からの解放を示す．

i）発側から，BYE リクエストを受信したシグナリングゲートは，切断（REL）メッセージにマッピングし，電話網の着側に送出する．通常の呼解放では，REL の理由表示パラメータに“＃16：正常切断”を設定する

ii）なお，着側からの REL に対しては，BYE にマッピングする

おわりに

BB フォンに代表されるインターネット事業者による IP 電話サービスが多くの加入者を獲得するなか，NTT コミュニケーションズや NEC といった大手も追随する形でサービスを開始した．このような状況から，今後，電話網との接続も本格化していくものと予想される．IP 電話と既存電話網との接続におけるメッセージのマッピングに関しては，すべてが厳密に決まっているわけではなく，また，SIP 仕様自体も詳細まで決まっておらず実装によるところが多いなど，現在では，相互接続性には多くの課題がまだあるといえる．今後は相互接続性の確保に向けた国内の標準化，相互接続テストなどが必要となってくる．

いずみ・としかつ　NTT アドバンステクノロジ(株)

好評発売中

CD-ROM 版 Interface2002

Interface 編集部 編　CD-ROM（Windows 用）
専用ビニール・ケース（A5 判）定価 13,000 円（税込）
ISBN4-7898-3776-9

CQ出版社　〒170-8461 東京都豊島区巣鴨 1-14-2　販売部 TEL.03-5395-2141　振替 00100-7-10665

Information ―ソフィアシステムズ，シンビアンのプラチナパートナープログラムに参加
(株)ソフィアシステムズは，シンビアン(株)のパートナープログラム「Symbian Platinum Partner Program」に参加すると発表した．この提携により，ソフィアからシンビアン OS 向けのソフトウェアソリューションが提供されることになる．

特集第3章

VoIPのシグナリングプロトコル SIPを用いたシグナリングの実際

中村一貴/水田栄一/四方涼子

VoIPにおいて相手との通話を確立するためには，通話相手のIPアドレスの特定から使用する音声CODECの指定まで，さまざまなパラメータを決定しなければならない．そのための手順をシグナリングといい，従来多く使われてきたH.323と，本章で解説するSIPがある．とくにSIPは，プロトコルがテキストベースであることから，実装が容易でデバッグもしやすく，近年普及が進んでいる．

そこでここではシグナリングの実例として，SIPを用いたシグナリングについて解説を行う．

（編集部）

VoIPのシグナリングにはいくつかの方式があるが，ここでは最近注目されているSIP（Session Initiation Protocol）についての説明を行う．

1 SIPの概要

● SIPの特性

SIPはテキストベースのプロトコルである．HTTPやSMTPなどの特徴を基に考え出されているため，リクエスト，レスポンス，アドレス表記などが非常に似ている．現在のSIPはIETF[注1]内のSIPワーキンググループでドラフト版の立案が進められており，当初はRFC2543[注2]として改訂を重ねてきたが，最新ののはRFC3261である．SIPはセッションの確立や切断を行うプロトコルであり，ピアツーピアの通信方式をとるが，制御方式としてはクライアント-サーバ型である．そのため，通信を行うSIPエレメントはクライアント（UAC[注3]）としてリクエストを送り，サーバ（UAS[注4]）としてレスポンスを行う．SIPエンドポイントといわれているすべてのSIPエレメントには，その両方が備わっている必要がある．

SIPは，IPネットワークで使用されるさまざまなエンドポイントとのセッションを確立することを目標としたプロトコルであり，取り扱うメディア（音声，映像，データなど）の対象を決めているわけではない．既存のVoIPの考え方では呼[注5]制御サーバがなければエンドポイント同士は呼接続できないが，SIPではピアツーピア通信に基本を置いているため，宛先の位置情報（contact address）がわかっていれば，SIPサーバを介することなく呼を接続することができる．

SIPではユーザーアドレス（SIP URI，後述）とユーザーのcontact addressとが分かれて対応付けされている．ユーザーアドレスで呼を扱うことにより，エンドユーザーはそのユーザーアドレスがどのcontact addressと対応付けされているのかを意識せずに通信できる．

● H.323との比較

現在広く使用されているVoIPプロトコルの一つにH.323がある．H.323は既存のPSTN[注6]でのシグナリングの考え方を基にした通信方式であるのに対して，SIPはIPネットワーク上でのプロトコルの考え方を基にして発案された通信方式である．

H.323はPSTNとの接続性に優れているが，基本的には音声や映像しか扱えない．また，既存のVoIP機器はAPIを用いてプログラミングされることが多く，その実装仕様が開発各社で違うことから，企業向けでの用途が主であった．

SIPは前項で述べている特性から，IPネットワーク上で広く使用されているHTTPやSMTPなどとの親和性に優れ，さまざまなサービスの拡張が行える．また，テキストベースなので，実装仕様を意識することなくアプリケーションや機器を開発できる．そのため，同一メーカーで統一するなどの制約を受けず，個人用途としても広くIPネットワーク上で使用することができる．

SIPは使用するメッセージ数が少ないのも特性の一つである．そのため，H.323に比べて手順が少なく，接続確立までの時間の短縮が図れる．また，メッセージ数が少ないことはそれにともなうエラーが発生する機会も減少するため，サービスの品質も向上するという利点がある．

● SIP URI

SIPでは，通常はSIP URI（Uniform Resource Identifiers）を用いてアドレス指定を行う．これは`sip:user@host`というメ

注1：IETF　：Internet Engineering Task Force，IPネットワーク上で開発されるさまざまな新しい技術の標準化を促進するために設立された組織．
注2：RFC　：Request for Comment，IETFが発行するドキュメント，IPネットワークの標準を決める文書と認められている．
注3：UAC　：User Agent Client，SIPのリクエストを生成する役割をもつ．
注4：UAS　：User Agent Server，SIPのレスポンスを生成する役割をもつ．
注5：呼　　：Call，利用者が通信を目的として通信回線を用いる動作．
注6：PSTN：Public Switched Telephone Networks，電話網．

ールアドレスに似た形式となり，その表現方法はパラメータの追加なども含めると多様である．ユーザーはSIP URIを利用することで，自分や通信相手の位置情報を参照できるようになる．

以降に代表的な形式と意味，アドレス例を示す．

例）sip:user:password@host:port;uri-parameters

● user

宛先となるhostにおける特定のエレメントの識別子．SIP URIに"@"が存在する場合，userフィールドを空にしてはならない．宛先となるホストが電話番号を処理することが可能なテレフォニーゲートウェイの場合，ここには電話番号を指定することができる．

● password

userに関連付けられたpasswordを指定することができる．しかし，暗号化されていないパスワードを送受信するということはセキュリティ上問題があるため，使用については推奨されない．

URIのuserinfoはこのuserフィールド，passwordフィールドおよびそれに続く"@"で構成される．URIのuserinfoはオプションであり，省略することができる．

● host

宛先のSIPエレメントを管理するドメインまたはネットワークアドレス．ドメイン形式を使用することが推奨される．

● port

リクエストの送信先ポート番号．

URIのhostportはこのhostフィールド，portフィールドで構成される．URIのhostportは必須だが，portフィールドはデフォルト(5060)ポートであれば省略することができる．

● URI parameters

そのURIから構築されるリクエストに関するパラメータ名とその値．URI parametersは，hostportコンポーネントの後に";"で区切られて追加される．下記にURI parametersの形式を示す．

parameter-name = parameter-value

parameter-nameにはtransport，maddr，ttl，user，method，lrなどがある．URI parametersは拡張可能であり，SIPのエレメントは未知のパラメータが付加されていた場合には無視する．

● PSTNとの接続事例

本来IPネットワーク上でのエンドポイント間接続を目的として立案されたSIPだが，PSTN網内へ発呼する場合も非常に容易である．SIP対応の「テレフォニーゲートウェイ」と呼ばれているIPエンドポイントを介して行う．SIPではPSTN電話番号に似たuser文字列（例：sip:1234@skywave.ne.jp）を使用できることから，区別するには，パラメータとしてuser=phoneを追加すべきである．

そのほかにも，RFC2806で定義されているtel URIを利用することも可能である．

〔リスト1〕リクエストラインの例

```
INVITE sip:sample@sip.ne.jp SIP/2.0
   ①        ②                  ③

①メソッド：INVITE
②Request-URI：sip:sample@sip.ne.jp
③プロトコルバージョン：SIP/2.0
```

2 SIPシグナリング

● SIPシグナリング

セッション確立のために，制御情報のやり取りを示したもので，RFC3261(SIP)，RFC2327(SDP：Session Description Protocol)などに規定されている．ここでは，SIPメッセージの構成要素であるメソッド，ヘッダ，SDPの各部，リクエストメッセージに対する応答（レスポンスメッセージ）について説明する．また補足として，音声や映像の伝達に使われるRTP(Real-time Transport Protocol)，エンドユーザーでサービスロジック（呼転送，通話拒否など）を作成可能なCPL(Call Processing Language)について説明する．

● SIPメソッド

リクエストの種別を表したもので，メッセージの最初の行に示されている．リクエストの最初の行は，リクエストライン(Request-Line)と呼ばれ，メソッド，Request-URI，プロトコルバージョンの順で構成される（**リスト1**）．Request-URIには，次のエレメントのURIが入る．メソッドによりその役割は異なる（**表1**）．RFC3261では，INVITE/ACK/BYE/CANCEL/REGISTER/OPTIONSの六つが規定され，それ以外のメソッドについては，拡張仕様として他のRFC(RFC3262，RFC3265など)に規定されている．

● ヘッダ

SIPメッセージのリクエストラインの次の行から記述されるも

〔表1〕SIPメソッド一覧

メソッド名	内　容	リファレンス
INVITE	セッションの開始	RFC3261
ACK	セッションの確立	RFC3261
BYE	セッションの終了	RFC3261
CANCEL	進行中セッションの取り消し	RFC3261
REGISTER	contact addressの登録	RFC3261
OPTIONS	サーバ-クライアント(UA)への機能の問い合わせ	RFC3261
INFO	セッション内でのシグナリング情報	RFC2976
PRACK	暫定レスポンスに対する確認要求	RFC3262
SUBSCRIBE	ユーザーの情報伝達要求	RFC3265
NOTIFY	ユーザー情報（ステータス）の伝達	RFC3265
UPDATE	セッション情報の更新	RFC3311
MESSAGE	インスタントメッセージの送信	RFC3428
REFER	呼転送	draft-ietf-sip-refer-07

New Products — IARシステム，ARM用コンパイラのC++およびマルチバイトサポートを発表
IARシステムズ(株)は，同社のARM用組み込みアプリケーション統合化開発環境「IAR Embedded Workbench for ARM」の新バージョンがC++テンプレート，日本語コードを含む複数バイトコードなどの新機能をサポートしたと発表した．

〔表2〕ヘッダー覧

Accept	Content-Encoding	Min-Expires	Route
Accept-Encoding	Content-Language	MIME-Version	Server
Accept-Language	Content-Length	Organization	Subject
Alert-Info	Content-Type	Priority	Supported
Allow	CSeq	Proxy-Authenticate	Timestamp
Authentication-Info	Date	Proxy-Authorization	To
Authorization	Error-Info	Proxy-Require	Unsupported
Call-ID	Expires	Record-Route	User-Agent
Call-Info	From	Reply-To	Via
Contact	In-Reply-To	Require	Warning
Content-Disposition	Max-Forwards	Retry-After	WWW-Authenticate

のである(**表2**). ヘッダは, ヘッダ名, “：”, ヘッダ値という順に記述され, 複数のヘッダ値がある場合には“；”で区切られる. ここでは, 頻繁に使用されるTo, From, Via, Contact, Call-ID, CSeq, Record-Routeの各ヘッダについて説明する(**リスト2**).

● To

Toヘッダには, リクエストの着信先URIが含まれる. Toヘッダでは, “<”“>”内にSIPのURIを含むことで, 表示名を利用できる.

例) To: test <sip:test@sip.ne.jp>

● From

Fromヘッダには, リクエストの発信元のアドレスが含まれ, Toヘッダ同様, 表示名を使うことができる. Fromヘッダでは, Tagパラメータが付加されることがあり, ダイアログ[注7]を識別するために使用される.

例) From: "test" <sip:test@sip.ne.jp>;tag=9a7fd854-32b3-440f-b8

● Via

Viaヘッダは, リクエストを転送するSIPサーバによって付加されるもので, 転送されるごとにヘッダが追加される. よって, Viaヘッダから経路情報を知ることができる. Viaヘッダは, SIPのバージョン, 転送プロトコル(UDP), サーバのIPアドレスまたはホスト名, ポート番号の順に記述される.

例) Via: SIP/2.0/UDP 10.0.0.2:5060

● Contact

Contactヘッダは, メッセージを生成する端末のcontact addressが含まれ, SIPサーバを経由せずに, 端末間で直接メッセージの送受信を行うことを可能にする. これにより, SIPサーバの負荷軽減にもつながる.

例) Contact:<sip:10.0.0.2:5060>

● Call-ID

Call-IDヘッダは, SIPセッションを識別するためにそれぞれ一意なIDが振られ, グローバルな識別を可能とするため, その直後に, “@”とIPアドレスが付加される.

例) Call-ID: c5c37d94-c959-45cd-a55f-eec4be2e6e15@10.0.0.2

〔リスト2〕ヘッダ例

```
Via: SIP/2.0/UDP 10.0.0.2:5060
From: "test" <sip:test@sip.ne.jp>;tag=9a7fd854-32b3-440f-b8
To: sip:info@sip.ne.jp
Call-ID: c5c37d94-c959-45cd-a55f-eec4be2e6e15@10.0.0.2
CSeq: 1 INVITE
Contact:<sip:10.0.0.2:14711>
User-Agent:Skywave SIP Packetel
Content-Type:application/sdp
Content-Length: 280
```

● CSeq

CSeqヘッダは, 数字とメソッド名から構成され, 同一のSIPセッションにおいて, 新たなリクエストごとにインクリメントされる. 発信元からのINVITEにより通話確立後, BYEメッセージを送出した場合には, CSeqナンバは2となる. ただし, ACKとCANCELメッセージに関しては, 加算の対象とはならない.

例) CSeq: 1 INVITE

● Record-Route

Record-Routeヘッダは, 特定のSIPサーバを必ず介してシグナリングを行う場合に使用される. とくに, セキュリティの管理やセッション(切断など)を把握する必要のある場合に使われ, Routeヘッダも同様の意味をもつ.

● レスポンス

受信したメッセージ(たとえば, INVITEメッセージ)に対する応答を示すもので, 3桁の数字からなるステータスコードが付加されている. ステータスコードの大まかな種類については, 六つのクラスで構成され(**表3**), はじめの5クラスについては, HTTPから引用したものである. SIPレスポンスは, SIPリクエストとは異なり, 先頭の行にステータスライン(Status-Line)が付き, プロトコルバージョン, ステータスコード, レスポンスフレーズの順で構成される(**リスト3**). とくに, ステータスコードが100番台のものは暫定レスポンスと呼ばれ, リクエスト処理が継続中(試行中/呼出中など)であることを示し, 200番台以降

注7：ダイアログ：しばらく(通話中)の間継続するUA間のセッションのステート状態. 通常はCall-IDによって保持されるが, UAが個別に付加するtag情報などによってグローバルに一意なものとなる.

New Products ―メトロワークス, CodeWarrior for Motorola HC12/HCS12 Ver2.0を発売
メトロワークスは, CodeWarrior for Motorola HC12/HCS12 Ver2.0の出荷を開始した. http://www.metrowerks.com/MW/download/default.aspより, 無償のSpecial Editionをダウンロードできる.

のものは最終レスポンスと呼ばれ，リクエスト処理の終了を示す．

● SDP

テキストベースの記述で，IPネットワーク上のマルチキャストセッションを意図して作成された．現在RFC2327として規定され，SIPのメディアネゴシエーションを行ううえで重要な役割を担っている．

SDPは，SIPメッセージボディ部に記述され(**リスト4**)，IPアドレス(**リスト4**の※1)，Port，メディアタイプ(例：音声/映像，※2)，メディアフォーマット(例：RTP[注8]/PCM，※3)などが含まれている．各パラメータには，必須とオプションがあり(**表4**，**リスト5**)，必須パラメータ内に，SIPにおいて使用されない記述も若干含まれる．また，いったん，セッションが確立した後でメディアを変更する場合には，再度INVITEを発行(Re-INVITE)することで変更することができる．

● CPL

IETFのIPTELワーキンググループによって規定された，XML(eXtensible Markup Language)ベースのサービスロジックである．CPLの大きな特性は，エンドユーザー側からSIPサーバに対しアップロードをすることで，エンドユーザー側のサービスを個別に設定することができることである．代表的なサービスに，時間指定/話中による転送機能やある特定の発信者からの着信規制などがある．例として，発信者番号が"0990"の場合には拒否するなどがある(**リスト5**)．

3 SIPエレメント

SIPでは，下記の各種のエレメントが定義されている．概要は前章でも説明してあるが，ここではB2BUAについても説明する．

● UA

ユーザーエージェント(User Agent)．SIPネットワーク内で

〔表3〕ステータスコード一覧

コード	内　容	説　明
1xx	暫定	リクエスト処理継続中
2xx	成功	リクエスト処理の成功
3xx	リダイレクト	リクエストの転送要求
4xx	クライアントエラー	リクエストにエラーが含まれる
5xx	サーバエラー	サーバエラーによりリクエスト未処理
6xx	グローバルエラー	リクエスト処理失敗(再試行なし)

注：1xxは，100～199の間を指す．

例．代表的なステータスコード
100：Trying
180：Ringing
200：OK
300：Multiple Choices
404：Not Found
500：Server Internal Error
604：Does Not Exist Anywhere

〔リスト3〕レスポンスの例

```
SIP/2.0 200 Ok
 ①        ②  ③

①プロトコルバージョン：SIP/2.0
②ステータスコード：200
③レスポンスフレーズ：Ok
```

〔リスト4〕SIPメッセージ

```
INVITE sip:info@sip.ne.jp SIP/2.0
Via: SIP/2.0/UDP 10.0.0.2:5060
From: "test" <sip:test@sip.ne.jp>;tag=9a7fd854-32b3-440f-b8
To: sip:info@sip.ne.jp
Call-ID: c5c37d94-c959-45cd-a55f-eec4be2e6e15@10.0.0.2
CSeq: 1 INVITE
Contact:<sip:10.0.0.2:14711>
User-Agent: Skywave SIP Packetel
Content-Type:application/sdp
Content-Length: 280
                                  } SIPヘッダ部
v=0
o=test 0 0 IN IP4 10.0.0.2
s=session
c=IN IP4 10.0.0.2 ――※1
b=CT:1000
t=0 0
m=audio 5004 RTP/AVP 97 0 8 4 101 ――※2
a=rtpmap:97 red/8000
a=rtpmap:0 PCMU/8000 ――※3
a=rtpmap:8 PCMA/8000
a=rtpmap:4 G723/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-16
                                  } SIPボディ部=SDP
```

```
注：SIPメッセージボディ部にSDPが含まれる。

  頻繁に使用されるパラメータのc/m/aについての説明は，次の通りである．

※1： 1 c=IN IP4 10.0.0.2
         ①  ②    ③
① IN：          ネットワーク種別(IPネットワーク)
② IP4：         アドレス種別(IPバージョン4)
③ 10.0.0.2：    IPアドレス

※2： m=audio 5004 RTP/AVP 97 0 8 4 101
         ①      ②     ③        ④
① audio：        メディアタイプ(音声)
② 5004：         ポート番号
③ RTP/AVP：      トランスポートプロトコル(Real time Transport
                        Protocol.  using the Audio/Video profile)
④ 97,0,8,4,101：ペイロードタイプ
                 (五つの値は，それぞれ下記のaパラメータのペイロードタイプ値を指す)

※3： a=rtpmap:0 PCMU/8000
          ①     ②    ③
① rtpmap：       RTP/AVPリスト
② 0：            ペイロードタイプ
③ PCMU/8000：    エンコード名(PCM μ-Law)/クロック(8000Hz)
                      (ITU G.711 PCM μ-Law Audio 64kbps)
```

注8：RTP：Real-time Transport Protocolの略．エンドポイント間で音声・映像などのリアルタイムなデータ伝送を目的に開発されたプロトコル(RFC1889)で，SIPメッセージのようなテキストベースではなく，ビットパターンで表現される．データ伝送にはUDPが使用され，IPネットワーク上を行き交う他のパケットと同じように扱われる．RTPはアプリケーション層のプロトコルであるため，QoS(Quality of Service)を保証するものではないが，RTCP(RTP Control Protocol)を使うことにより，パケットの損失，伝送遅延，不連続パケットなどの障害を検知することができる．とくに，RTCPのパケットタイプであるSR(Sender Reports)/RR(Receiver Reports)パケットで，RTPパケットのレポートの送受信ができ，SDES(Source Description)パケットでは，セッションの参加者情報(セッション番号，ユーザー名，電話番号，電子メールなど)が伝送される．

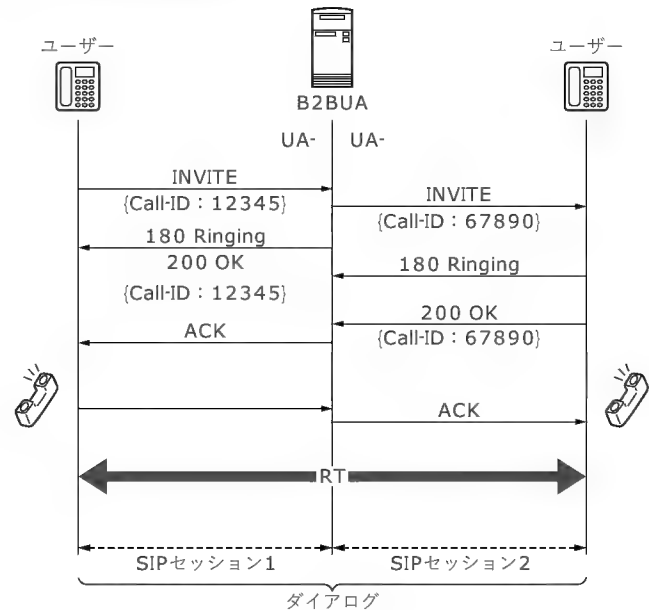
Information ―ウェブソフト・インターナショナルとメトロワークスが組み込みミドルウェアビジネスで協業
ウェブソフト・インターナショナル(株)とメトロワークス(株)は，携帯情報機器向け組み込みブラウザEspritとGUIフレームワークPowerPartsの販売で協業することを発表した．

〔表4〕セッション記述一覧

パラメータ	名　前	必須/オプション
v	プロトコルバージョン	必須
o	発信元	必須
s	セッション名	必須
t	セッション時刻	必須
m	メディアタイプ	必須
c	コネクション情報	オプション
i	セッション情報	オプション
u	URI記述	オプション
e	メールアドレス	オプション
p	電話番号	オプション
b	周波数帯情報	オプション
z	タイムゾーン調整	オプション
k	暗号キー	オプション
a	セッション属性	オプション
r	リピート回数	オプション

〔リスト5〕CPLの例

```
<?xml version="1.0" ?>
  <!DOCTYPE cpl PUBLIC "-//IETF//DTD RFCxxxx CPL 1.0//EN" "cpl.dtd">
  <cpl>
    <outgoing>
      <address-switch field="original-destination" subfield="tel">
        <address subdomain-of="0990">
          <reject status="reject"
                  reason="Not allowed to make Q2 calls." />
        </address>
      </address-switch>
    </outgoing>
  </cpl>
```

〔図1〕B2BUAの概略

のエンドポイントとなる．ハードウェア型やソフトウェア型などがある．テレフォニ-ゲートウェイはIPネットワーク上ではエンドポイントであるため，ここに含まれる．UAC(User Agent Client)とUAS(User Agent Server)の両方が備わっているのは前述した．また，後述するB2BUAもUAの一種になる．

● プロキシサーバ

SIPメッセージ中の`Request-URI`を参照して，そのアドレスへ呼を転送するのがおもな目的だが，`Request-URI`のドメイン名またはネットワークアドレスが管理下の場合は，その規定に基づいて動作する．セッションの保持の仕方によりステートレス，コールステートフル，トランザクションステートフル，実装仕様によりルースルータなどの種類がある．

● リダイレクトサーバ

`Request-URI`に対するcontact addressを探索し，その情報を転送レスポンス(3xx)で発信元に返す．発信元はこのレスポンスメッセージ中のcontact addressに`INVITE`リクエストメッセージを送る．

● レジストラ

ユーザー登録を行う．`REGISTER`リクエストメッセージを受け取り，データベースへcontact addressを格納する．

● ロケーションサーバ

レジストラが，`REGISTER`リクエストメッセージによって得られたユーザーのcontact addressやその他のユーザー情報を格納するデータベースである．

● B2BUA

バックツーバック ユーザーエージェント(B2BUA)とは，一種のSIPエンドポイントであるが，UAを二つもつ(**図1**)．

発呼者とUA-1との間，UA-2と着呼者との間には別々のSIPセッションが確立されるが，ダイアログは保持される．

B2BUAはSIP端末であるために，SIP UAの規定以外には動作に関する明確な定義は必要とされない．

ユーザーAはUA-1と，ユーザーBはUA-2とそれぞれCall-IDの違う独立したSIPセッションのやり取りを行っている．

B2BUAは，tagの情報などからこの独立したSIPセッションが互いに対だということを把握し，対応させているため，セッション1でのセッション情報は必ずセッション2に反映され，ステートが維持される．

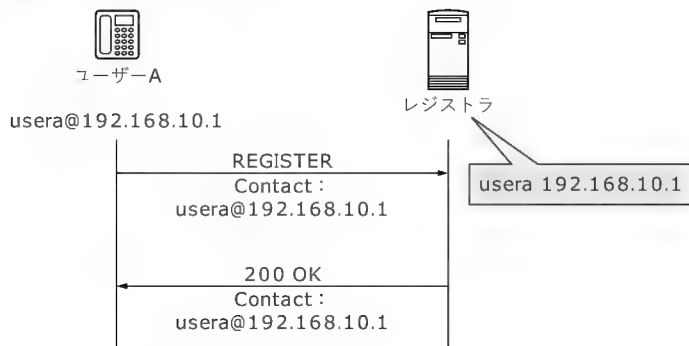
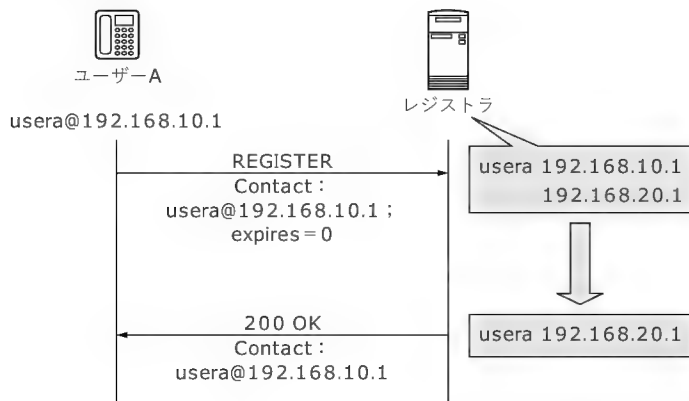
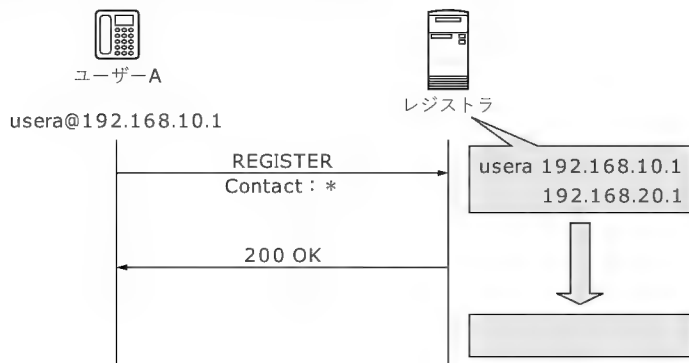
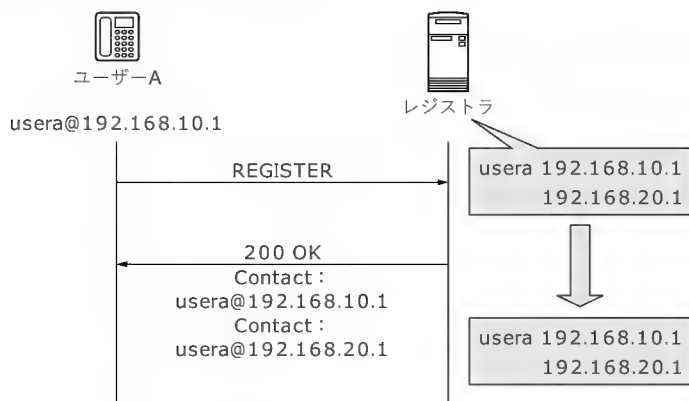
ステートフルプロキシサーバが存在する場合のダイアログに似ているが，プロキシサーバが扱うダイアログは一つのSIPセッションである．

4 代表的なSIPコールシーケンス

● 登録(REGISTER)

登録することによって，ロケーションサーバのデータベース内に`address-of-record URI`(ユーザーのパブリックアドレス)がcontact addressとされる．このcontact addressは使用するSIP端末のアドレスであり，複数登録できる．`To`ヘッダに`address-of-record URI`を含むリクエストメッセージをプロキシサーバが受けたとき，メッセージは対応付けされている

Information ―モトローラ，メトロワークス，ACCESSが組み込みリナックスOS向けトータルソリューションの提供で提携
モトローラ(株)，ACCESS(株)，メトロワークス(株)は，マルチメディア機能とBluetooth機能などを搭載したドラゴンボールi.MX(innovative Media eXtention)，NetFront v3.0，統合開発環境CodeWarriorをベースとしたトータルソリューションを提供していくことで合意した．

〔図2〕contact addressの追加

〔図3〕contact addressの削除(その1)

〔図4〕contact addressの削除(その2)

〔図5〕contact addressの問い合わせ

contact addressに転送される．

レジストラから返ってくるレスポンスメッセージには，現在登録されているすべてのcontact addressが含まれる．また，Contactヘッダに“q”パラメータを追加することで，複数のcontact addressに優先度をつけることもできる．

REGISTERリクエストメッセージを使用して行える処理に，下記のものがある．

● contact addressの追加(**図2**)

レジストラに送られるREGISTERリクエストメッセージのContactヘッダには，新たに対応付けされるcontact addressが含まれる．また，address-of-record URIはToヘッダに含まれる．

● contact addressの削除(**図3**，**図4**)

削除を行いたい場合には，対応するcontact addressに対する有効期限を0に設定したREGISTERリクエストメッセージを送る．また，すべてのcontact addressの削除を行いたいときには，Contactヘッダの値を“*”に設定したREGISTERリクエストメッセージを送る．

すべてのcontact addressの削除を行った際，返ってくるレスポンスメッセージにはContactヘッダが含まれていない．

● contact addressの問い合わせ(**図5**)

現在登録されているcontact addressを問い合わせたい場合には，Contactヘッダを含まないREGISTERリクエストメッセージを送る．

● 発呼から正常接続および切断まで(INVITE，BYE)

代表的な接続として下記の4種類がある．

● 端末間を直接接続する場合での呼の確立と切断(**図6**)

① ユーザーAが192.168.0.11にいるユーザーBに話したいとする．ユーザーAはユーザーBの端末のIPアドレスを知っているので，ユーザーAの端末のSDPを含んだINVITEリクエストメッセージを直接ユーザーBの端末に送る．このときのRequest-URIはsip:192.168.0.11である

② ユーザーBの端末はINVITEリクエストメッセージを受け，180 RingingをユーザーAに返すとともに，着信処理(着信音を鳴らすなど)を行う

③ ユーザーAの端末は180 Ringingを受け，呼び出し音を鳴らす

④ ユーザーBが通話を許可すると，ユーザーBの端末のSDPを含んだ200 OKがユーザーAに返される

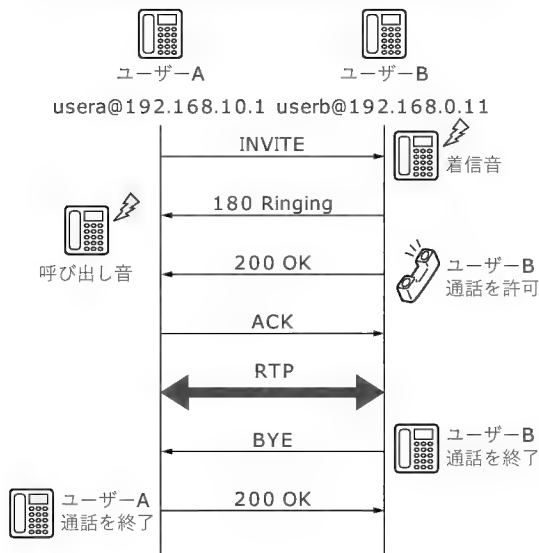
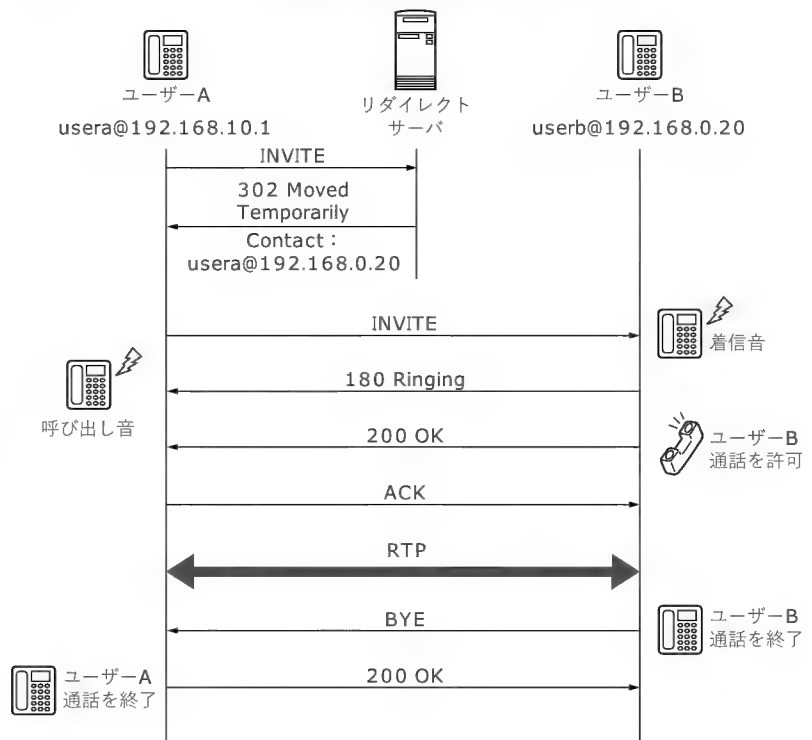
⑤ ユーザーAの端末は200 OKを受けると，ACKメッセージをユーザーBに返す

⑥ 以上の結果，ユーザーAとユーザーBの端末間で双方向のRTPセッションが確立される

⑦ ユーザーBが通話を終了するとユーザーBの端末はRTPセッションを切断し，BYEリクエストメッセージをユーザーAに送る

⑧ ユーザーAの端末はBYEリクエストメッセージを受け，200

Information ―ケイデンス，三菱電機に論理合成ツールをOEM提供
日本ケイデンス・デザイン・システムズは，三菱電機(株)が米国ケイデンス・デザイン・システムズ社と，ケイデンス製のSoC設計用論理合成ツールの2年間のOEM契約を締結したことを発表した．

〔図6〕端末間を直接接続する場合での呼の確立と切断

〔図7〕リダイレクトサーバが存在する場合での呼の確立と切断

OKをユーザーBに送る

●リダイレクトサーバが存在する場合での呼の確立と切断(**図7**)

① ユーザーAがsip:userb@skywave.ne.jpで登録されているユーザーBに話したいとする．ユーザーAはユーザーAの端末のSDPを含んだINVITEリクエストメッセージをリダイレクトサーバに送る

② リダイレクトサーバが，ロケーションサーバにユーザーBのcontact addressを確認すると，sip:userb@192.168.0.20に登録されていることがわかった．そこで，リダイレクトサーバはユーザーAに302 Moved Temporarilyを返す．このレスポンスメッセージのContactヘッダに登録されているcontact addressが含まれる

③ ユーザーAの端末はそのcontact addressにINVITEリクエストメッセージを送り，その後の処理は直接接続と同じようになる(上記参照)

●ステートレスプロキシサーバが存在する場合での呼の確立と切断(**図8**)

① ユーザーA(sip:usera@skywave.ne.jp)がユーザーB(sip:userb@skywave.ne.jp)に話したいとする．ユーザーAはユーザーAの端末のSDPを含んだINVITEリクエストメッセージを自分のドメインのプロキシサーバに送る

② プロキシサーバはユーザーBのcontact addressをロケーションサーバから取得すると，そのcontact addressにINVITEリクエストメッセージを転送する．転送の際，プロキシサーバはViaヘッダに自分のアドレスを追加する．また，ユーザーAに対して処理中であることを知らせるために，100 Tryingを送る

〔図8〕ステートレスプロキシサーバが存在する場合での呼の確立と切断

③ ユーザーBの端末はINVITEリクエストメッセージを受け，180 Ringingを返すとともに，着信処理を行う．この180 RingingはViaヘッダにプロキシサーバのアドレスが追加されているため，プロキシサーバに返される

Information — Project JXTAのダウンロード数が100万を突破
サン・マイクロシステムズは，サンのWebサイトからProject JXTAをダウンロードした開発者の数が100万人を超えたとを発表した．JXTAは，オープンソース方式で標準化されたピアツーピア(P2P)技術．

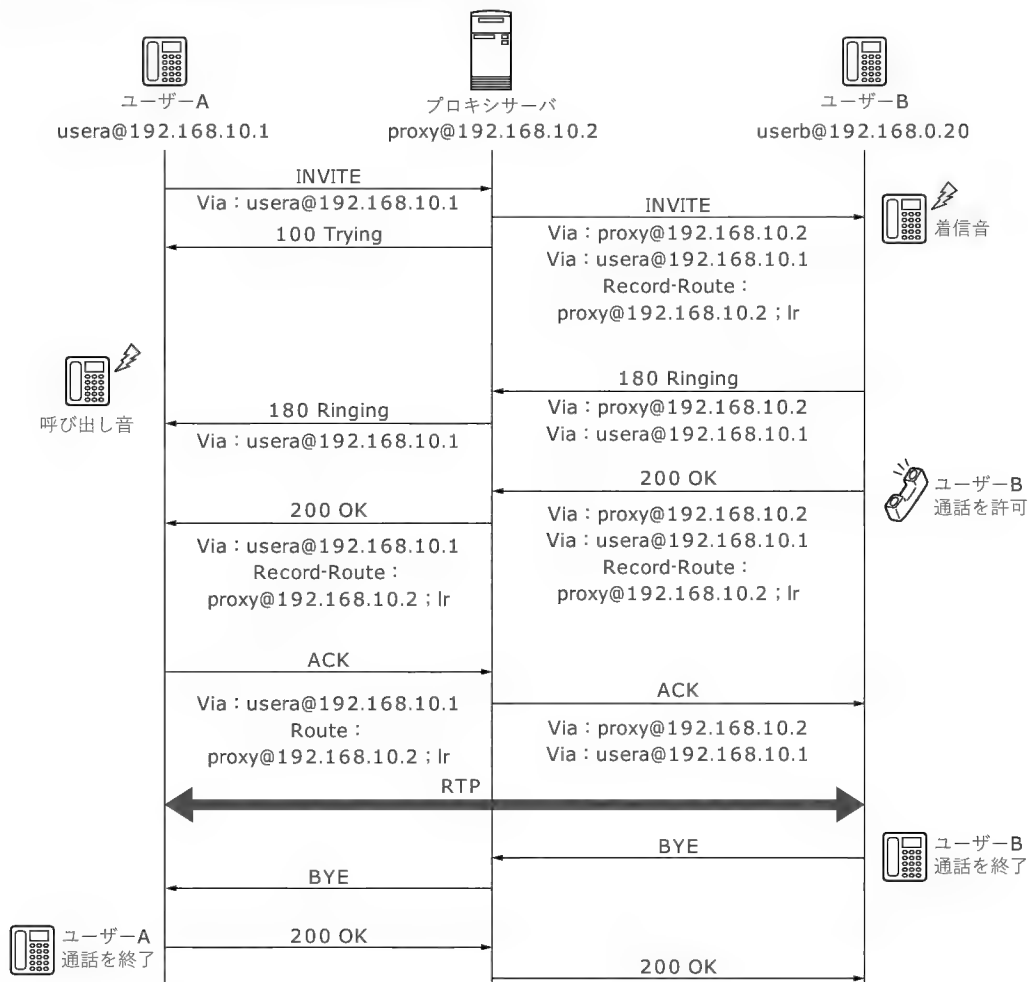
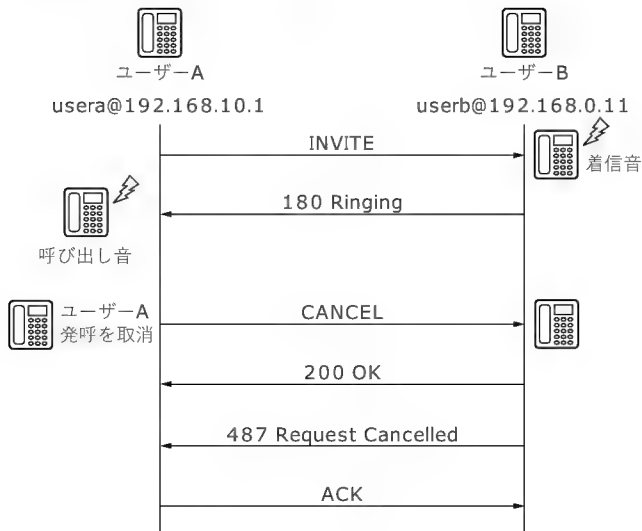
④ プロキシサーバは180 Ringingを受け，Viaヘッダから自分のアドレスを削除するとユーザーAに転送する
⑤ ユーザーAの端末は180 Ringingを受け，呼び出し音を鳴らす
⑥ ユーザーBが通話を許可するとユーザーBの端末のSDPを含んだ200 OKが返される．この200 OKはViaヘッダにプロキシサーバのアドレスが追加されているため，プロキシサーバに返される
⑦ プロキシサーバは200 OKを受け，Viaヘッダから自分のアドレスを削除するとユーザーAに転送する
⑧ ユーザーAの端末は200 OKを受けると，ユーザーBのcontact addressをメッセージのContactヘッダから取得し，ACKメッセージを直接ユーザーBに返す
⑨ 以上の結果，ユーザーAとユーザーBの端末間で双方向のRTPセッションが確立される
⑩ ユーザーBが通話を終了するとユーザーBの端末はRTPセッションを切断し，BYEリクエストメッセージを直接ユーザーAに送る
⑪ ユーザーAの端末はBYEリクエストメッセージを受け，200 OKをユーザーBに送る

●ステートフルプロキシサーバが存在する場合での呼の確立と切断(**図9**)

① ユーザーA(sip:usera@skywave.ne.jp)がユーザーB(sip:userb@skywave.ne.jp)に話したいとする．ユーザーAはユーザーAの端末のSDPを含んだINVITEリクエストメッセージを，自分のドメインのプロキシサーバに送る
② プロキシサーバはユーザーBのcontact addressをロケーションサーバから取得するとそのcontact addressにINVITEリクエストメッセージを転送する．転送の際，プロキシサーバはViaヘッダに自分のアドレスを追加する．また，以降，このセッション内のすべてのメッセージが自分を通過するようにRecord-Routeヘッダに自分のアドレスを追加する．そして，ユーザーAに対して処理中であることを知らせるために100 Tryingを送る
③ ユーザーBの端末はINVITEリクエストメッセージを受け，180 Ringingを返すとともに，着信処理を行う．この180

〔図9〕ステートフルプロキシサーバが存在する場合での呼の確立と切断

Information ― ウェアラブルPHS電話機「WRISTOMO」にCompact NetFrontを採用
(株)ACCESSは，(株)エヌ・ティ・ティ・ドコモが発表した腕時計型ウェアラブルPHS電話機「WRISTOMO」にACCESSのマイクロブラウザCompact NetFrontが採用されたことを発表した．

〔図10〕CANCELリクエストメッセージ

〔図11〕Re-INVITEリクエストメッセージ

Ringing はViaヘッダにプロキシサーバのアドレスが追加されているため，プロキシサーバに返される

④ プロキシサーバは180 Ringingを受け，Viaヘッダから自分のアドレスを削除するとユーザーAに転送する

⑤ ユーザーAの端末は180 Ringingを受け，呼び出し音を鳴らす

⑥ ユーザーBが通話を許可するとユーザーBの端末のSDPを含んだ200 OKが返される．その際，プロキシサーバのアドレスを含むRecord-Routeヘッダが含まれている．この200 OKはViaヘッダにプロキシサーバのアドレスが追加されているため，プロキシサーバに返される

⑦ プロキシサーバは200 OKを受け，Viaヘッダから自分のアドレスを削除するとユーザーAに転送する

⑧ ユーザーAの端末は200 OKを受けると，Record-Routeヘッダから取得したプロキシサーバのアドレスにlrが付いているため，値をACKメッセージのRouteヘッダとして追加すると共に，プロキシサーバに返す

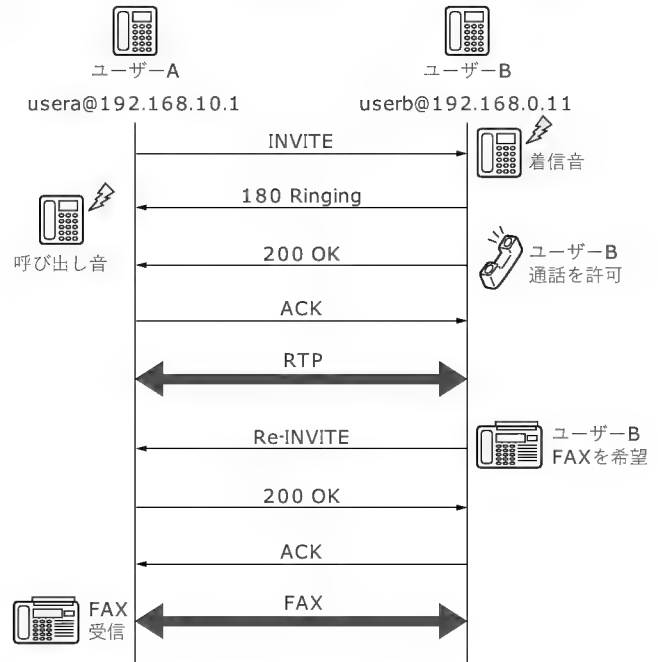
⑨ プロキシサーバはACKメッセージを受け，Routeヘッダから自分のアドレスを削除するとユーザーBに転送する

⑩ 以上の結果，ユーザーAとユーザーBの端末間で双方向のRTPセッションが確立される

⑪ ユーザーBが通話を終了すると，ユーザーBの端末はRTPセッションを切断し，BYEリクエストメッセージを直接ユーザーAに送る

⑫ ユーザーAの端末はBYEリクエストメッセージを受け，200 OKをユーザーBに送る

● 発呼の取り消し(CANCEL)

CANCELリクエストメッセージは，すでに送られたINVITEリクエストメッセージを取り消すために使用する(**図10**)．

CANCELリクエストメッセージを受け取ったSIPエレメントはいっさいの処理を終了し，INVITEリクエストメッセージに対して487 Request Terminatedを返す．

ただし，CANCELリクエストメッセージを使用して取り消すことができるのは200番台以降の最終レスポンスメッセージを返されていないINVITEリクエストメッセージだけである．また100番台の暫定レスポンスメッセージを返されていない場合には，100番台のレスポンスを受け取るまでCANCELリクエストメッセージの送信を待つ．

● セッションの変更(Re-INVITE)

SIPでは，すでに確立されたセッションを変更することができる．変更内容としては，アドレスやポート番号の変更，メディアストリームの追加，削除などが考えられる．このような変更の手段としてセッションを確立したダイアログ内で新たなINVITEリクエストメッセージを送る方法が使われる．このようなINVITEリクエストメッセージをRe-INVITEリクエストメッセージと呼ぶ(**図11**)．Re-INVITEリクエストメッセージはどちらの端末からも送信できる．

● インスタントメッセージ(MESSAGE)

MESSAGEは，テキストなどを相手に送るために使用されるメソッドで，一般的にインスタントメッセージ(IM)といわれている．SIP端末では，MESSAGEリクエストメッセージを受け，レスポンスメッセージとして200 OKを返す．また，MESSAGEは，セッションが確立していなくても送信できるという利点がある(**図12**)．

● プレゼンス(SUBSCRIBE, NOTIFY)

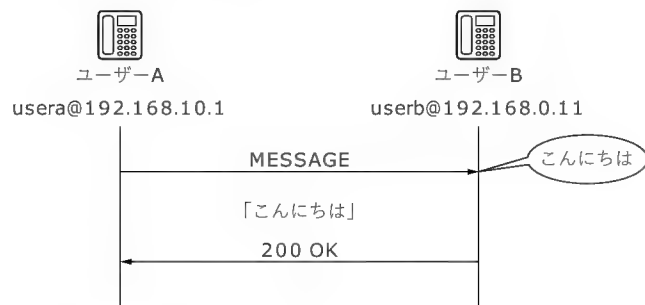
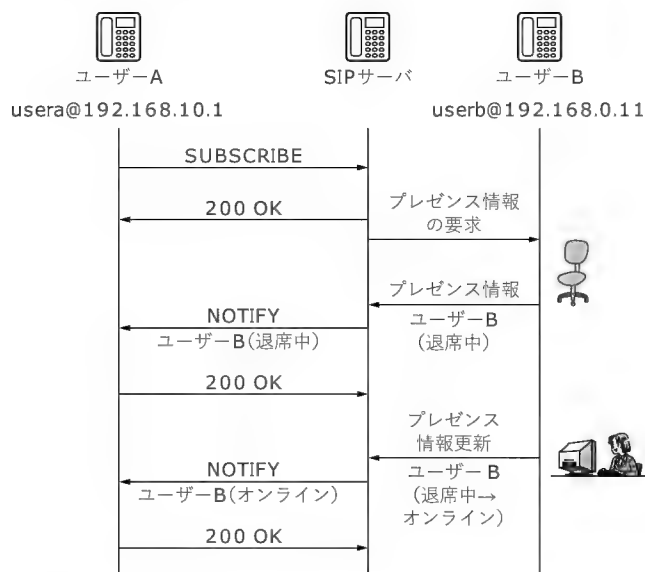
SIPでは，イベント通知をするために，SUBSCRIBEと

New Products — LCDパネル制御用16ビットマイコン
富士通(株)は，富士通デバイス(株)と共同で，AV機器およびディジタル家電向けに，最大192セグメントのLCDパネル制御機能を搭載した16ビットマイコン「MB90800シリーズ」を発売した．価格は¥900～．

NOTIFYというメソッドがある．一般的に，プレゼンス機能といわれ，現在相手がどのようなステータスなのかを知ることができるとともに，ステータスの変化についても通知される．イベント通知を要求するメソッドがSUBSCRIBEで，イベント通知に使われるメソッドがNOTIFYである．相手のステータスを知るためにまず，SUBSCRIBEリクエストメッセージを送る．SUBSCRIBEリクエストメッセージを受けた端末はレスポンスメッセージとして200 OKを返し，NOTIFYリクエストメッセージを送る．NOTIFYリクエストメッセージを受けた端末は200 OKを返す．SIP端末はステータス変更があったとき，即座にNOTIFYリクエストにて，ステータス変更を通知する(**図13**)．

とくに，複数のステータス(オンライン，オフライン，電話中，退席中など)を一覧として見るリストのことを，Buddy List(バディリスト)という．

現在，上記のSIPをベースとしたインスタントメッセージ，プレゼンス機能については，IMPP(Instant Messaging and Presence Protocol)ワーキンググループのSIMPLE(SIP for Instant Messaging and Presence Leveraging Extensions)にて規定されている．

〔**図12**〕**インスタントメッセージ**

〔**図13**〕**プレゼンス**

5 サービス例

このシステムはページャ(ポケベル)網を使用して，PDAをウェイクオン(自動起動)し，呼を着信できるようにするシステムである(**図14**)．

通常PDAの端末は電池の消費を抑えるため，使用していない間は電源を切ってある場合がほとんどである．そのため呼を着信できず，電話としての活用が難しい．そこで，ページャ網を使用して電源が切られているPDAでも呼を着信できるようにしたのがこのシステムである．

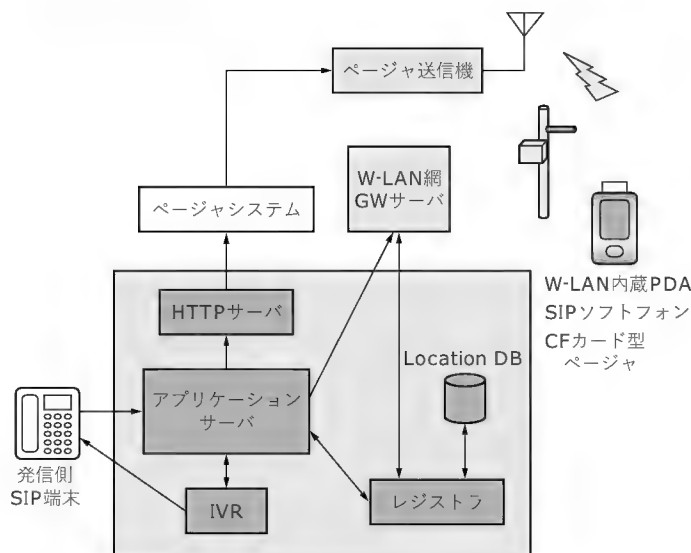
このシステムの利点としては，外部への接続をインターネット網を使用して行っているため，設置場所の制限を受けずにどこにでもサービスを提供できることである．また，ページャシステムに送るデータはプロトコルとしてHTTPを使用しているので，対応したページャ機能をもつPDAさえあれば，どのようなページャシステムとも接続することが可能である．

- システム構成
- アプリケーションサーバ
- HTTPサーバ
- IVR(音声自動応答システム)
- レジストラ
- SIPソフトフォン
- コールフロー

① 発信者が発信端末のSDPを含んだINVITEリクエストメッセージがアプリケーションサーバに送る

② アプリケーションサーバは100 Tryingを発信端末に返した後，レジストラにContactヘッダを含まないREGISTERリクエストメッセージを送って着信者の登録状態を確認する

〔**図14**〕**システムの一例**

Information ── 富士通とAMDがフラッシュメモリ事業を統合
富士通(株)とAMDは，両社のフラッシュメモリ事業を統合再編し，米国に新会社「FASL Limited Liability Company」を設立することについて，基本合意書を締結した．出資比率は富士通40%，AMD 60%．

③ レジストラからのレスポンスメッセージにContactヘッダが含まれない場合，登録なしとみなしてアプリケーションサーバよりHTTPサーバへ必要なデータ（発信者ID，着信者IDなど）を送る．また，IVRにINVITEリクエストメッセージを転送し，レスポンスメッセージを発信端末に返してセッションを確立する．このセッションでは発信者に音声アナウンス（例：“ただいま，おつなぎしております”）を流す
④ HTTPサーバからページャシステムへデータを転送する
⑤ ページャシステムは受け取ったデータを基に，PDAに装着されたCFカード型ページャに発呼する
⑥ CFカード型ページャは着信後，PDAのウェイクオンを行う
⑦ SIPソフトフォンは，無線LAN経由でレジストラにPDA端末のIPアドレスを含んだREGISTERリクエストメッセージを送って登録する
⑧ レジストラに登録されると，アプリケーションサーバより着信端末へ発信端末のSDPを含んだINVITEリクエストメッセージが送られる
⑨ 着信者が着信を許可すると，IVRへ呼の切断を促すBYEリクエストメッセージを送る．また，発信端末には着信端末のSDPを含んだRe-INVITEリクエストメッセージが送られる
⑩ 発信端末からのレスポンスメッセージを着信端末に転送してセッションを確立する

おわりに

SIPは，現在もIETFのSIPワーキンググループで標準化の策定中である．オプションなどについては単独でワーキンググループが組織されているものもあり，その外郭も随分とはっきりしてきた．サービスとしても，海外の電話サービス会社では，すでにすべての音声通信をSIPにて行っている事業者も出現している．

最初にも述べたが，SIPはプロトコルであり通信の内容を限定しているものではない．現在はVoIPとしての利用が主であるが，それ専用の通信方式ではないので，音声や映像通信に限られずその他のアプリケーションを利用してさまざまなサービスが構築可能である．

サービスが展開されるにしたがい，音声通信としての利用にとどまらず，通信方法がSIPだということを意識せず利用できるようになるであろう．

なかむら・かずたか/みずた・えいいち/よも・りょうこ
スカイウェイブ（株）

COMPUTER TECHNOLOGYシリーズ　好評発売中

ハードリアルタイム機能を使いこなす
RTLinuxテキストブック

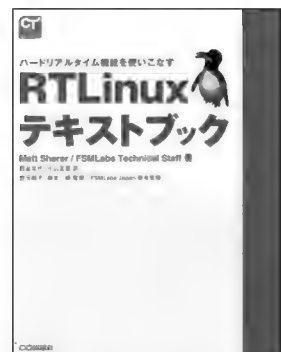
Matt Sherer／FSMLabs Technical Staff 著
西谷年代／小山友里 訳　吉元純子／森友一朗 監修
FSMLabs Japan 総合監修
B5変型判　164ページ　定価2,940円（税込）
ISBN4-7898-3705-X

本書は，RTLinuxの開発元であるFSMLabsが書き下ろした解説書に，より読者に便利な情報を追加・再編集した本です．RTLinuxをより便利に，そしてより深く使いこなすための情報が記されています．すでにRTLinuxを使っている読者も，これから使おうと考えている読者も，本書からプログラミングを行ううえでの有益な情報を得られます．

第I部　RTLinuxの基本
第1章　RTLinuxへの招待
第2章　リアルタイムシステム，コンセプト，およびRTLinux
第3章　カーネルモジュールの入門
第4章　RTLinuxモジュール
第5章　RTLinux API
第6章　LinuxとRTLinuxの情報伝達
第II部　サンプルプログラム
第7章　カーネルモジュールの“Hello World”
第8章　RTLinuxマルチスレッドモジュール
第9章　スケジューリングジッタの測定
第10章 LinuxからRTLinuxへの割り込み信号
第11章 リアルタイムとユーザー空間の間の共有メモリ
第III部　Appendix
Appendix A　略語リスト
Appendix B　RTLinuxバージョン3 API
Appendix C　システムテスト
Appendix D　RTLinuxで拡張された関数群

CQ出版社　〒170-8461 東京都豊島区巣鴨1-14-2　販売部 TEL.03-5395-2141　振替 00100-7-10665

Information ― 日立システムとミラクル・リナックス，Linux HAクラスタで業務提携
（株）日立システムアンドサービスとミラクル・リナックス（株）は，MIRACLE LINUX上におけるLinux HAクラスタ製品「CLUSTERPRO」とOracleデータベースを組み合わせたミッションクリティカルソリューションを共同で展開することに合意した．

特集第4章

品質の向上とデータレートの低減が鍵となる VoIPで用いられる音声CODECの詳細

青木 実

音声データをディジタル化して流すには，規定されたフォーマットにしたがって音声を変換する必要があり，これをCODECという．

CODECは音声圧縮技術と密接な関係がある．VoIPでは限られたネットワーク帯域を有効に活かすため，音声データをエンコードすることが一般的だ．通常のVoIPで用いられる音声圧縮技術には，対象を人間の音声に特化することにより，通常の音声圧縮よりも大幅な圧縮率を誇るものなどがある．

そこで本章では，VoIPにおける音声品質の向上の要であるCODECについて解説を行う．

（編集部）

はじめに

VoIPで通話を行うためには，音声をディジタル信号として取り扱う必要があります．A-D変換器とD-A変換器さえあれば，とりあえずディジタル化された音声をネットワーク上に流すことはできますが，情報量が多すぎてすぐに破綻してしまうでしょう．

ネットワークの限られた帯域を効率良く利用するためには，音声データの情報量を圧縮伸張する"音声CODEC"が必要となります．

CODECとは「エンコーダ（符号器）＋デコーダ（復号器）」という意味の造語です．エンコーダは非圧縮の音声データから符号化されたデータを出力します．逆にデコーダは，符号化されたデータから非圧縮の音声データを復元します．A-D変換器やD-A変換器のことをCODECと呼ぶ場合もありますが，ここでは非圧縮音声データと符号データの変換部分を指すものとします（**図1**）．

音声CODECの場合，一般的なデータ圧縮（ZIPやLHAなど）と異なり，完全に元と同じデータは復元できません．音声が劣化するのと引き換えに，高い圧縮率を得ることに主眼を置いているからです（最近の音楽用CODECでは完全に復元するものもある）．音声CODECは入出力がディジタルデータなので，中身はすべて数値演算処理（ディジタル信号処理）になります．演算処理はハードウェアでもソフトウェアでも実現可能ですが，処理内容が非常に複雑なため，ソフトウェアでなければ実現できないようなものもあります．ディジタル信号処理演算を高速に行うためのDSP（Digital Signal Processor）も多く使われます．

〔図1〕CODECの範囲

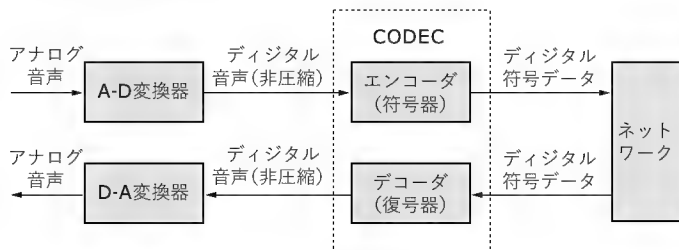



表1に主要な音声CODECを示します．

本章では，まず最初に音声CODECの比較や選択を行う場合に，考慮すべきポイントについて説明します．

次に音声CODECの原理について説明します．とくにVoIPで使用されることが多いITU-T G.723.1やG.729 Annex Aの基礎となっているCELP方式について詳しく解説します．また，RTP（Real-time Transport Protocol）についても説明します．

最後にITU-T勧告のG.711，G.723.1，G.726およびG.729 Annex Aの使用方法を説明します．

音声CODEC比較のポイント

音声CODECを使用したシステムを構築する場合に検討すべきポイントを紹介します．

音声CODECの品質を一つの尺度だけで評価するのは難しく，また危険なことでもあります．使用環境やネットワークの状態を含めた総合的な判断が必要です．

● 音声帯域・サンプリング周波数

音声CODECの分類上，サンプリング周波数が8kHzのものを狭帯域（Narrow Band）CODEC，16kHzのものを広帯域（Wide Band）CODECと呼んでいます（**表2**）．

従来の電話（PSTN）の音声帯域は3.4kHzであり，会話を行うためにはこの程度の帯域があれば十分です．そのため，VoIPに使用される音声CODECは，ほとんどが狭帯域CODECに属しています（**表1**）．

● ビットレート，パケット長

符号データの情報量はビットレート（kbps）で表します．音声CODECのビットレートは，ほとんどが2kbpsから64kbpsの間です（**表1**，**図2**）．

VoIPでは符号データにIP/UDP/RTPのヘッダ情報が加わるので，それらの情報量を含めたビットレートを考慮する必要があります．たとえば，8kbpsの音声CODECを20msのパケット長で使用する場合，符号データの20バイトにIPヘッダの20バイト，UDPヘッダの8バイト，RTPヘッダの12バイトが加算

〔表1〕主要な音声CODEC

標準化組織・勧告番号	名称・方式	サンプリング周波数(kHz)	ビットレート(kbps)	フレーム長(ms)	原理遅延(ms)	品　質(MOS)	おもな用途
ITU-T G.711	PCM	8	64	―	0.125	4.10	ISDN
ITU-T G.723.1	MP-MLQ/ACELP	8	6.3/5.3	30	37.5	3.9/3.65	VoIP
ITU-T G.726	ADPCM	8	16/24/32/40	―	0.125	3.85	PHS
ITU-T G.728	LD-CELP	8	16	0.625	0.625	3.85	
ITU-T G.729	CS-ACELP	8	8	10	15	3.92	PDC
ITU-T G.729 Annex A	CS-ACELP	8	8	10	15	3.7	VoIP
ARIB STD-27	PSI-CELP	8	3.45	40	45		PDC
ARIB STD-27	VSELP	8	6.7	20			PDC
ARIB STD-27	ACELP	8	6.7	20	25		PDC
3GPP GSM 06.10	RPE-LTP	8	13	20		3.5	GSM FR
3GPP GSM 06.20	VSELP	8	5.6	20		3.5	GSM HR
3GPP GSM 06.60	ACELP	8	12.2	20	25		GSM EFR
TIA IS-54	VSELP	8	7.95	20			D-AMPS
TIA IS-641	ACELP	8	7.4	20	25		D-AMPS
3GPP TS 26.071	AMR	8	4.75/5.15/5.9/6.7/7.4/7.95/10.2/12.2	20	20/25		3G
INMARSAT	IMBE	8	4.15	20			船舶電話
DDVPC FS-1016	CELP	8	4.8	30	105		軍用
DDVPC FS-1015	LPC-10e	8	2.4	22.5	90		軍用
DDVPC	MELP	8	2.4	22.5	45.5		軍用
ITU-T G.722	SB-ADPCM	16	48/56/64	0.125	1.5		
3GPP TS 26.171/ITU-T G.722.2	AMR-WB	16	6.6/8.85/12.65/14.25/15.85/18.25/19.85/23.05/23.85	20	25		

〔表2〕音声帯域による分類

分　類	サンプリング周波数	音声帯域
狭帯域(Narrow Band)	8kHz	3.4kHz
広帯域(Wide Band)	16kHz	7kHz

されて合計60バイトのパケットになります．これは24kbpsのビットレートに相当します．同じCODECでもパケット長を40msにするとオーバヘッドが軽減されて16kbps相当になりますが，当然ながら遅延が増えてしまいます．

● VAD/DTX/CNG

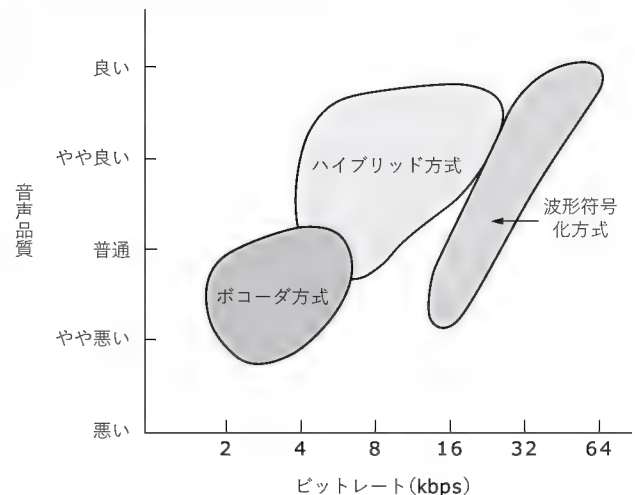
平均的なビットレートを下げるための技術で，無音圧縮とも呼ばれます．まずエンコーダで音声の有無を判断し(VAD)，無音の場合には背景雑音情報だけを送信するか，またはまったく送信せず(DTX)，デコーダでは無音時の背景雑音を擬似的に生成します(CNG)．これらの略語の意味は以下のとおりです．

VAD ：Voice Activity Detection(有音/無音検出)
DTX ：Discontinuous transmission(不連続送信)
CNG ：Comfort Noise Generation(雑音生成)

● 音声品質の評価

音声CODECの評価にはMOS(Mean Opinion Score)が多く使われます．MOSは，複数の人が実際に音を聞いて評価をした結果を統計的にまとめたもので，1(最低)から5(最高)までの点数で表します．

● 背景雑音の影響・音声以外の信号

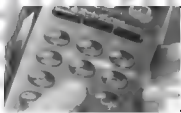
人間の発声構造をモデル化したCODEC(ボコーダ方式，ハイブリッド方式)は，人間の音声以外の信号で品質が劣化します．とくにビットレートが低い場合に顕著です．

〔図2〕音声CODEC方式による分類

音楽も通したい場合には，波形符号化方式か高ビットレートのハイブリッド方式のCODECを使用すべきです．

背景雑音が大きな場所では，ノイズキャンセラを併用すると効果があります．

● タンデミング

エンコードとデコードを繰り返し行うことをタンデミングといい，繰り返すほど品質は劣化します．送話者と受話者で同じCODECを使用し，途中で無駄な変換を行わないようなシステ

New Products ― IARシステムズ，組み込み用Bluetoothプロトコルスタックの新バージョンを発売
IARシステムズ(株)は，同社の組み込み用IAR Bluetoothプロトコルスタックのv1.40を発売した．RAM 2Kバイトで動作し，SIMアクセスプロファイルをサポートしたことが特徴．

ム設計が望まれます．逆にある程度の品質劣化を容認すれば変換装置（ゲートウェイ）を経由して，異なるCODEC同士でも通話ができます．

● フレームエラー

VoIPではRTP/UDPが使用されるため，ネットワークの都合でパケットが破棄されてしまうことがあります．通常，1パケットには20～200ms相当の符号データが入っているため，パケットロスは大きな品質劣化につながります．

フレーム単位で処理を行うCODECには，フレームエラーに対応したものがあります．デコーダに「フレームエラーが発生した」という情報を入力するだけで，過去の符号データから違和感の少ない音声を合成することができます．

サンプル単位で処理を行うCODECでは，G.711 Appendix Iが利用できます．これは過去に再生した音声を保存/分析し，パケットエラーが発生したときに違和感の少ない音声を作り出す方法です．

● 遅延量

フレーム単位で処理を行うCODECでは，フレーム長の入力音声がそろうまでは処理が開始できません．また入力音声の分析やフィルタ処理のためにCODEC内部で固有の遅延が発生する場合があります．このように原理的に避けられない遅延を原理遅延と呼びます．その他の遅延の要素としては処理遅延，パケット化による遅延，伝送遅延，ジッタバッファによる遅延などがあります．

● ジッタ

VoIPではジッタ（＝遅延のゆらぎ）の影響を取り除くために，ジッタバッファを使用します．ジッタの低減は本来ネットワークで対応すべき問題ではありますが，音声CODECの特徴や機能を利用することにより，遅延や音質劣化が少ないジッタバッファを構成することができます．

サンプル単位で処理を行うCODECではパケット単位ではなく，サンプル単位のよりきめ細かいジッタバッファの制御を行うようにします．

フレーム単位で処理を行うCODECでは，前述の「フレームエラー対応機能」を積極的に使うことにより，過度の遅延増加を抑えるようにします．

〔表3〕音声CODECの標準化組織

組織名	URL
国際電気通信連合（ITU-T）	http://www.itu.int/ITU-T/
（財）日本ITU協会	http://www.ituaj.jp/
社団法人情報通信技術委員会（TTC）	http://www.ttc.or.jp/
（社）電波産業会（ARIB）	http://www.arib.or.jp/
米国電気通信産業協会（TIA）	http://www.tiaonline.org/
欧州通信規格協会（ETSI）	http://www.etsi.org/
3GPP	http://www.3gpp.org/
The Department of Defense Digital Voice Processor Consortium（DDVPC）	http://maya.arcon.com/ddvpc/

● 実現方法

これまでの組み込み装置では，DSPや専用LSIが使用されることが多かったのですが，最近では，汎用プロセッサが使用される事例も増えています．このことから，パソコンやPDAで使用される機会も増えてきています．

標準化されているCODECは各標準化組織からソースコードを入手することができます．しかし，それらはアルゴリズムの解説を目的としており，そのままでは十分な性能（演算速度）が得られません．使用するプロセッサに合わせて最適化を行う必要があります．とくにITU-T G.723.1やG.729などはbasopという固定小数点DSP向きの演算ライブラリで記述されており，汎用プロセッサでは大きなオーバヘッドになります．

● 標準化

不特定な人々が通話をするオープンなシステムでは，標準化された音声CODECが必要です．VoIPではITU-T（国際電気通信連合）勧告のG.711，G.723.1，G.726，G.729 Annex Aなどが使用されます．

表3に音声CODECの標準化を行っている組織をまとめました．これらの組織から勧告書やソースコードを入手することができます．ITU-T勧告については，日本ITU協会やTTCから日本語に翻訳されたものを入手することもできます．

今回は紹介できませんでしたが，標準化されていない音声CODECも多数ありますので，通話範囲が限られたシステムではさらに選択肢が広がります．

● 特許

標準化されているCODECの中にも，使用するためには特許料を支払わなければならないものがあります．製品開発を行う場合には注意が必要です．

音声CODECの原理

音声CODECの原理は，大きく次の三つの方式に分類することができます．これらの方式のビットレートと音声品質の関係を**図2**に示しました．

● 波形符号化方式

入力信号の波形そのものの復元を目的とした方式です．

サンプル単位で処理を行い，各サンプルの量子化ビット数を減らしたり，隣接サンプルの相関性を利用した圧縮を行います．

入力信号を人間の声に限定しないので，音楽や雑音などでも良好な品質が得られますが，他の方式と比べて高いビットレートが必要です．またビットレートを低くすると極端に品質が劣化するという欠点もあります．

ITU-T G.711やG.726がこの方式です．

● ボコーダ方式

人間の発声構造をモデル化した方式です．人間の声は肺から出た空気が声帯を振動させ，その振動（励振信号）が声道フィル

Information ── IBMとNVIDIA，戦略的提携を締結
NVIDIAとIBMは，両社の間で複数年の戦略提携が結ばれたことを発表した．これにより，NVIDIAのGeForce GPUはIBMによって製造されることになる．

タ(喉，鼻，唇など)を通って出てきたものです(**図3**)．声道フィルタは合成フィルタとも呼ばれます．ボコーダ方式のエンコーダでは有声音と無声音の区別，励振信号，声道フィルタ特性などをパラメータ化し，デコーダではそのパラメータから音声を合成します．

ボコーダ方式では波形符号化方式のようにサンプル単位の処理ではなく，複数サンプルを集めたフレーム単位で処理を行います．人間の声道の特性は5～30msごとに変化するので，それに相当するフレーム単位で分析を行う必要があるわけです．

低ビットレートでも音声が再生できるという利点はありますが，音声合成特有の不自然な音になりがちです．また，人間の声以外の信号では極端に品質が悪く，周囲雑音の影響を受けやすいという欠点もあります．

● ハイブリッド方式

波形符号化方式とボコーダ方式の長所を合わせた方式です．

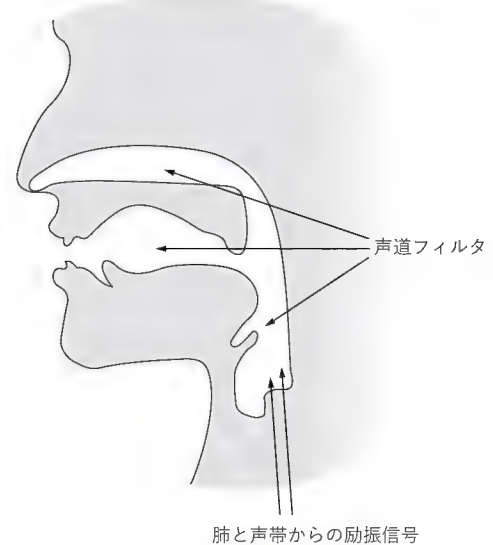
ボコーダ方式と同様な人間の発声構造をベースにしながら，入力信号の波形をできるだけそのまま復元するための工夫が施されており，低ビットレートでも波形符号化方式と同等な品質が得られます．

ボコーダ方式と同様に，人間の声以外の信号では品質が悪くなる傾向があります．また処理が非常に複雑になるという欠点もあります．しかし「低ビットレート」と「音声での良好な品質」は，会話を目的としたVoIPには最適な方式です．

ITU-T G.723.1やG.729はハイブリッド方式の音声CODECです．ハイブリッド方式をさらに細かく分類すると，励振信号の生成手法によってCELP，MPE(Multi Pulse Excitation：マルチパルス励振)，RPE(Regular Pulse Excitation：正規パルス励振)などの方式があります．

CELP方式の原理

前項で音声CODECの原理を簡単に説明しましたが，ハイブリッド方式の中でもVoIPで使用されることが多いCELP(Code-Excited Linear Prediction：符号励振線形予測)方式ついて，もう少し詳しい説明をします．

〔図3〕
人間の発声構造

ITU-T G.723.1(5.3kbps)やG.729は，このCELP方式を採用しています．

最初にデコーダと符号データについて説明します(**図4**)．デコーダは音声を生成するためのメカニズムそのものであり，符号データはデコーダの各コンポーネントへの情報源と捉えることができます．デコーダにおいてもっとも良い音声が再生できるような符号データを選択することが，エンコーダの役目になります．

● デコーダと符号データ

励振信号は適応コードブックと固定コードブックの信号に，それぞれゲインを掛けて加算することにより得られます．励振信号を合成フィルタに通すと音声が再生されます．

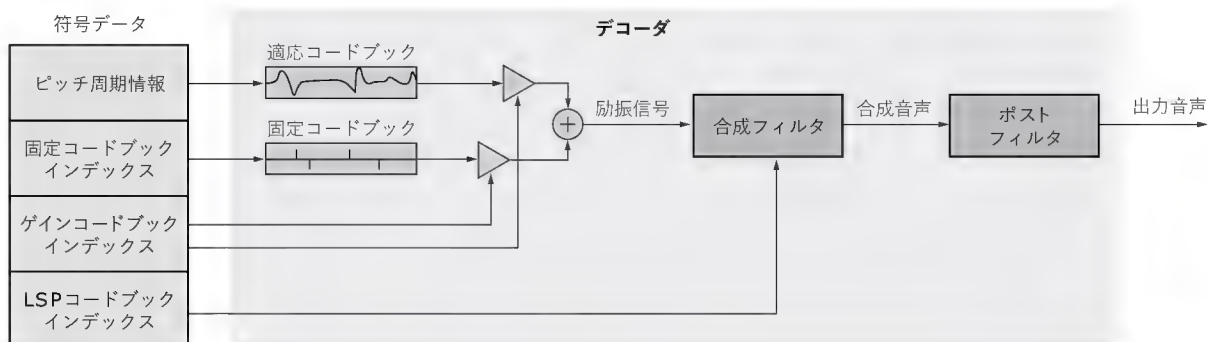
● 適応コードブック

適応コードブックの実体は，過去に生成した励振信号そのものです．音声信号は周期性が強いので，過去に生成した励振信号を保存しておき，ピッチ周期情報に基づいて再利用します．

● 固定コードブック

固定コードブックはあらかじめ決められた複数の波形データ

〔図4〕CELPデコーダ

Information ― モンタビスタとデジオン，ディジタル家電向けホームネットワーク・ソリューションで業務提携
モンタビスタソフトウエアジャパン(株)と(株)デジオンは，ディジタル家電向けホームネットワークソリューションで業務提携した．デジオンは，開発中のホームネットワークソリューション「DiXiM」のLinux用ミドルウェアをMontaVista Linuxへ対応させる．

の集まりです．そのインデックスが符号データになります．

とくにACELP（Algebraic CELP）の場合は固定コードブックとして振幅が一定のパルス（振幅情報がない）を使用するため，パルスの位置情報だけで励振信号が表現できます．そのため，コードブックのためのメモリが必要ありません．またエンコーダにおいては，代数的な特徴を利用して，少ない演算処理でコードブックの探索を行うことができます．

●**ゲインコードブック**

ゲインコードブックには，適応コードブックと固定コードブックのゲイン情報が両方とも含まれています．2種類のゲイン情報を組にしたものが複数用意されており，そのインデックスが符号データになります．

●**合成フィルタ**

通常，合成フィルタには線形予測フィルタがもちいられます．合成フィルタの情報として線形予測係数と相互変換が可能で，かつ符号化に適したLSP（Line Spectral Pair）パラメータが使用されます．複数のLSPパラメータがコードブックに用意されており，そのインデックスが符号データになります．

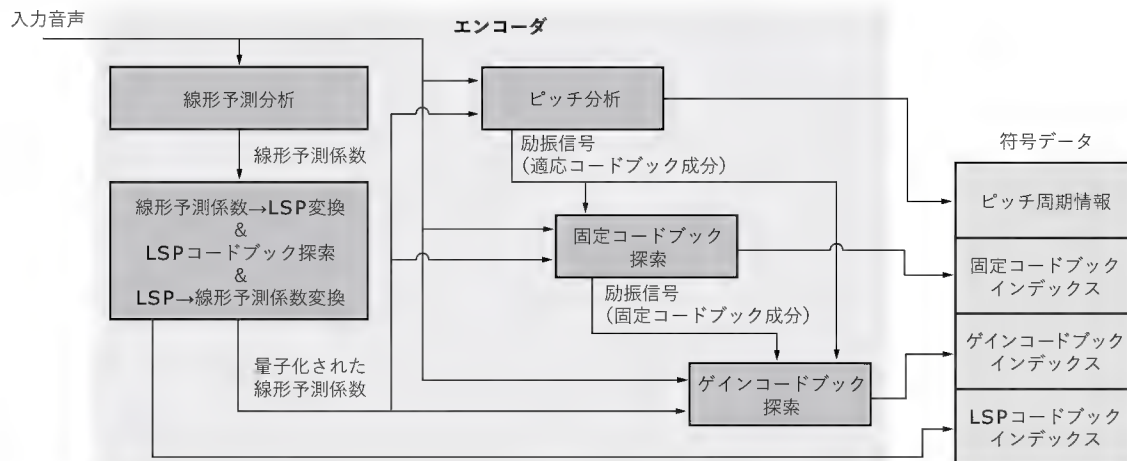
●**ポストフィルタ**

聴感上の品質を改善するためのフィルタです．ピッチやフォルマントを強調することにより，合成した音声のザラザラ感が低減されます．

● エンコーダ

図5にエンコーダの符号化手順を示します．

最初に入力音声信号の線形予測分析を行い，線形予測係数を求めます．これをLSPパラメータに変換し，もっとも近いものをLSPコードブックの中から選びます．さらに後のコードブック探索のために，選んだLSPパラメータを再び線形予測係数に変換しておきます．

次にピッチ分析，固定コードブック，ゲインコードブックの探索を順に行い，量子化された線形予測係数で合成した音声がもっとも入力信号に近くなるようなインデックスを求めます．

RTPについて

VoIPでは，符号化した音声データの伝送を行うためにRTP（Real-time Transport Protocol：リアルタイム転送プロトコル）を使用します．

● RTPの特徴

一般的なRTPパケットの構造を**図6**に示します．

RTPの大きな目的は符号データのフレーミング（パケット化）を行い，RTPヘッダのシーケンス番号とタイムスタンプ情報を利用することにより，送信側のパケット送出タイミングを受信側でも復元できるようにすることです．

伝送データのフレーミングを行うという点ではトランスポート層に近い役割を果たしますが，効率良くリアルタイム制御を行うためにアプリケーション層で直接フレーミングを行います．最適なフレーミング方法は符号データごとに異なるので，RTPでは詳細な仕様を定義していません．符号データごとの具体的なフレーミング方法，すなわちRTPパケットのフォーマットはIETFからインターネットドラフトとして公開されています（http://www.ietf.org/internet-drafts/draft-ietf-avt-profile-new-13.txt）．

RTPでは，下位の転送プロトコルにUDPを使用します．もしネットワークでパケットが破棄された場合でも，TCPのように再送せず，新しいパケットを転送し続けるほうがリアルタイム通信には適しています．

RTPには符号データを伝送する機能しかありません．RTPパケットの伝送制御や監視はRTCP（RTP Control Protocol：RTP制御プロトコル）により行います．

● RTPヘッダ

図7にRTPヘッダの構造を示します．

●バージョン（V）

RTPのバージョン番号．現在（RFC1889）は2です．

〔図5〕CELPエンコーダ

New Products ― 日本電気エンジニアリング，VoIPゲートウェイ「IPMASTER-1040シリーズ」を発売
日本電気エンジニアリング（株）は，VoIP SIPネットワークと既存電話網との接続を実現する，SIP対応VoIPゲートウェイ「IPMASTER-1040シリーズ」を発売した．音声圧縮方式として，G.711/G.729a/G.723.1に対応している．

●パディング(P)

パディングデータの有無を示します．

●拡張ヘッダ

拡張ヘッダの有無を示します．拡張ヘッダはPTごとに定義されており，存在する場合はCSRC識別子の後に挿入されます．

● CSRC識別子の数(CC)

●マーカー(M)

使用方法はPTごとに定義されます．

●ペイロードタイプ(PT)

伝送する符号データの種類を示します．

●シーケンス番号

初期値が乱数で，パケットごとに1ずつ増加します．受信側でパケットの破棄や受信順序の入れ替わりを検出することができます．

●タイムスタンプ

パケットに含まれるもっとも古いデータが発生した時刻を表します．クロックの周波数はPTごとに定義され，音声CODECの場合はサンプリング周波数になります．たとえば，8kHzサンプリングの音声コーデックを20msのパケット長で伝送する場合，1パケットには160サンプル分の音声データが含まれており，タイムスタンプもパケットごとに160ずつ増加します．

●同期送信元(SSRC)識別子

RTPパケットの送信元を識別するための識別子です．識別子の値には乱数を使用します．

●寄与送信元(CSRC)識別子

RTPでは複数の送信元から送られるパケットをミキシングする「ミキサ」という機能が定義されています．たとえば，2人の話者から送信される2系統の音声パケットをそれぞれデコードし，加算した後で再びエンコードすれば一つのRTPパケットが得られます．このような場合，CSRC識別子として元のパケットのSSRC識別子を列挙します．1対1の通信の場合はCSRC識別子は存在せずCCの値は0になります．

〔図6〕RTPパケット

IPヘッダ	UDPヘッダ	RTPヘッダ	符号データ

〔図7〕RTPヘッダ

<table>
<tr><th></th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>[0]</td><td colspan="2">V</td><td>P</td><td>X</td><td colspan="4">CC</td></tr>
<tr><td>[1]</td><td>M</td><td colspan="7">PT</td></tr>
<tr><td>[2]</td><td colspan="8" rowspan="2">シーケンス番号</td></tr>
<tr><td>[3]</td></tr>
<tr><td>[4]</td><td colspan="8" rowspan="4">タイムスタンプ</td></tr>
<tr><td>[5]</td></tr>
<tr><td>[6]</td></tr>
<tr><td>[7]</td></tr>
<tr><td>[8]</td><td colspan="8" rowspan="4">同期送信元(SSRC)識別子</td></tr>
<tr><td>[9]</td></tr>
<tr><td>[10]</td></tr>
<tr><td>[11]</td></tr>
<tr><td>[12]</td><td colspan="8" rowspan="4">寄与送信元(CSRC)識別子
(CCの数だけ列挙される．
まったくない場合もある)</td></tr>
<tr><td>[13]</td></tr>
<tr><td>[14]</td></tr>
<tr><td>[15]</td></tr>
</table>

ITU-T勧告の音声CODEC

ここからは，ITU-T勧告の各種音声CODECについて，例を示しながら解説を行います．

ITU-T G.711

PCM(Pulse Code Modulation：パルス符号変調)と呼ばれます．非圧縮のリニアPCMに対して対数圧伸PCMと呼ばれることもあります．

ISDN回線で通話を行うときの音声CODECがG.711です．アナログ回線でも交換機の間の通信はすべてディジタル化されており，そこで使われているのがG.711です．ですから，「G.711＝従来の電話音声」と考えて間違いありません．G.711にはμ-lawとA-lawという二つの方式が含まれており，日本の電話で使用されているのはμ-lawです．

● 原理

対数的に振幅の大きな信号ほど荒い量子化をすることにより情報量を圧縮します．人間の聴覚は，大きな音の歪みにはあまり敏感ではないことを利用しています．

● 入出力

サンプル単位の処理になります(**図8**)．

エンコーダもデコーダもμ-law/A-lawの選択を指定する必要があります．符号データにμ-law/A-lawの識別情報は含まれていません．

なお，μ-lawの音声データは14ビット精度ですが，フルスケール(－8192～8191)は使用できないので注意が必要です．

●音声データのフォーマット

14ビットリニアPCM，－8159～8159の値(μ-law)

13ビットリニアPCM，－4096～4095の値(A-law)

●符号データのフォーマット

8ビット/サンプル

● RTP

ペイロードタイプはμ-lawが「0」，A-lawが「8」です．

RTPでは20ms以上のパケット送出間隔が推奨されているので，1パケットを160サンプル以上の符号データで構成するようにします．

● 入手方法

G.711のCソースコードは，ITU-T G.191「Software tools for speech and audio coding standardization」に含まれています．

● G.711の拡張機能

G.711は本来サンプル単位の処理を行うCODECですが，VoIPのように複数サンプルをパケット化して使用するための拡張が行われています．

Information ― インターネット総合研究所と松下電工，IPv6対応HXの共同開発で提携
(株)インターネット総合研究所(IRI)と松下電工(株)は，IPv6に対応したネットワークゲートウェイHome eXchange(HX)の共同開発に関して提携した．

Appendix Iはパケットロス対策のための拡張です．デコーダにおいて過去に再生した音声データをバッファに蓄積し，パケットロスが発生したときにバッファのデータを使用して音声を合成します．

Appendix IIでは，VAD/DTX/CNGを使用する場合の背景雑音のレベルや周波数特性の符号化方法が規定されています．ただし，勧告にはVAD/DTX/CNGは含まれていないので，別途用意する必要があります．

これらの拡張は，G.726でも使用可能です．

ITU-T G.726

ADPCM（Adaptive Differential Pulse Code Modulation：適応差分パルス符号変調）と呼ばれます．

G.726では，音声データ入出力としてG.711符号データが使用されます．そのため，G.711で通話をしているネットワークの途中をG.726で置き換え，部分的にビットレートを節約することが簡単にできます．一般的に音声CODECはエンコードとデコード処理を繰り返すと音声品質が劣化しますが，G.726の場合は2回以上繰り返してもまったく劣化しません．したがって，G.711の通話系の複数箇所にG.726を挿入しても，1回分の劣化しか発生しません．

G.726 Annex Aでは音声データとしてリニアPCMを使用する方法を規定しています．

● 原理

エンコーダは過去の出力信号から入力信号を予測し，予測した入力信号と実際の入力信号との差分情報だけを伝送します．デコーダは過去の出力から予測した入力信号に，エンコーダから送られた差分情報を加えることにより音声を再生します．予測を行う部分では適応処理が行われています．ビットレートの違いは差分情報の量子化ビット数の違いです．

● 入出力

サンプル単位の処理になります（**図9**）．

エンコーダもデコーダもビットレートを指定する必要があります．符号データにはビットレートの情報は含まれていません．

●音声データのフォーマット

G.711符合データ（μ-lawまたはA-law）

14ビットリニアPCM（G.726 Annex A）

●符号データのフォーマット

5ビット/サンプル（40kbps）

4ビット/サンプル（32kbps）

3ビット/サンプル（24kbps）

2ビット/サンプル（16kbps）

● RTP

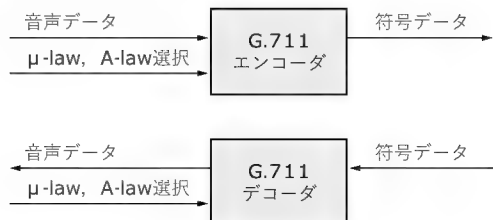
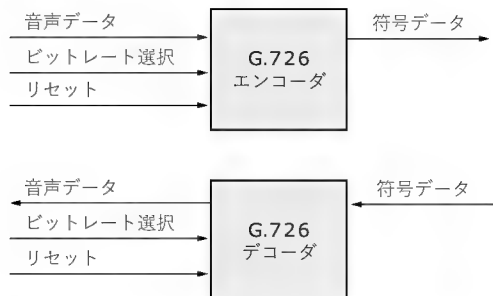
定められたペイロードタイプの番号はありません．ペイロードタイプは，シグナリングの過程で動的に割り当てられます．

1サンプルの符号データは2〜5ビットと半端な長さなので，RTPで通信を行うためにはオクテット（8ビット）単位になるように，複数サンプルをまとめる必要があります（**図10**）．

さらにパケット送出間隔は20ms以上が推奨されているので，160サンプル以上の符号データを一つのパケットにまとめます．

● 入手方法

G.726のCソースコードはITU-T G.191「Software tools for speech and audio coding standardization」に含まれています．

〔図8〕G.711の入出力

〔図9〕G.726の入出力

〔図10〕G.726符号データ

A，B，C……はそれぞれ1サンプルの符号データ．
( )内の数字は各符号データのビット番号（0がLSB）

	7 (MSB)	6	5	4	3	2	1	0 (LSB)
[0]	B(2)	B(1)	B(0)	A(4)	A(3)	A(2)	A(1)	A(0)
[1]	D(0)	C(4)	C(3)	C(2)	C(1)	C(0)	B(4)	B(3)
[2]	E(3)	E(2)	E(1)	E(0)	D(4)	D(3)	D(2)	D(1)
[3]	G(1)	G(0)	F(4)	F(3)	F(2)	F(1)	F(0)	E(4)
[4]	H(4)	H(3)	H(2)	H(1)	H(0)	G(4)	G(3)	G(2)

(a) 5ビット/サンプル，40kbpsの場合

	7 (MSB)	6	5	4	3	2	1	0 (LSB)
[0]	B(3)	B(2)	B(1)	B(0)	A(3)	A(2)	A(1)	A(0)

(b) 4ビット/サンプル，32kbpsの場合

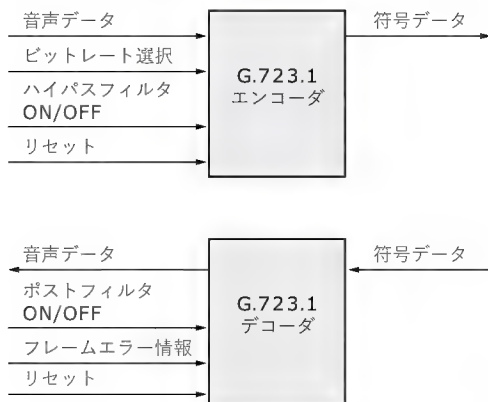
	7 (MSB)	6	5	4	3	2	1	0 (LSB)
[0]	C(1)	C(0)	B(2)	B(1)	B(0)	A(2)	A(1)	A(0)
[1]	F(0)	E(2)	E(1)	E(0)	D(2)	D(1)	D(0)	C(2)
[2]	H(2)	H(1)	H(0)	G(2)	G(1)	G(0)	F(2)	F(1)

(c) 3ビット/サンプル，24kbpsの場合

	7 (MSB)	6	5	4	3	2	1	0 (LSB)
[0]	D(1)	D(0)	C(1)	C(0)	B(1)	B(0)	A(1)	A(0)

(d) 2ビット/サンプル，16kbpsの場合

New Products — SHマイコンのJTAG用変換アダプタ「AUD to HUDI変換アダプタ」
京都マイクロコンピュータ（株）は，日立製SHマイコンのJTAGで採用されているAUD 36ピン用コネクタで，HUDI 14ピンも使用できる変換アダプタ「AUD to HUDI変換アダプタ」を発売した．価格は￥8,000．

〔図11〕G.723.1の入出力

ITU-T G.723.1

5.3kbpsのACELP(代数CELP)と6.3kbpsのMP-MLQ(Multi Pulse - Maximum Likelihood Quantizer：最ゆう[注1]量子化マルチパルス符号)のデュアルレートの音声CODECです.

二つのビットレートがありますが，内部の構造はほとんど共通で，フレーム単位にビットレートを切り換えることができます.

● 入出力

30msのフレーム単位の処理になります(**図11**).

エンコーダでビットレートの指定をします．ビットレート情報は符号データの中に含まれます.

エンコーダのハイパスフィルタとデコーダのポストフィルタをそれぞれ有効/無効にすることができます.

デコーダにはフレームエラー情報を入力することができます.

●音声データのフォーマット

16ビットリニアPCM，240サンプル/フレーム

●符号データのフォーマット

192ビット/フレーム(6.3kbps)

160ビット/フレーム(5.3kbps)

● RTP

ペイロードタイプは｢4｣です.

RTPでは，20ms以上のパケット送出間隔が推奨されていますが，G.723.1のフレーム長は30msなので，1フレームからパケットを構成できます．また，2フレーム以上の符号データを一つのパケットにまとめることもできます.

図12(次頁)と**表4**に1フレーム分の符号データのフォーマットを示します.

● 入手方法

ITU-TよりCソースコードが入手できます.

● G.723.1の拡張機能

G.723.1には，VAD/DTX/CNG機能を拡張したAnnex Aと，浮動小数点版のAnnex Bがあります(**表5**).

〔表4〕G.723.1符号データのパラメータ

パラメータ	内　容
LPC	LSP VQ index
ACL0	Adaptive Codebook Lag
ACL1	Differential Adaptive Codebook Lag
ACL2	Adaptive Codebook Lag
ACL3	Differential Adaptive Codebook Lag
GAINx	Combination of adaptive and fixed gains (xth subframe)
MPOS	The 4 MSB of POSx codewords are combined to form a 13-bit index
POSx	Pulse position index (xth subframe)
PSIGx	Pulse sign index (xth subframe)
GRIDx	Grid index (xth subframe)
UB	Unused bit

〔表5〕G.723.1拡張機能

	ビットレート(kbps)	演算精度	特　徴
G.723.1	5.3/6.3	固定小数点	
G.723.1 Annex A	5.3/6.3 (DTX)	固定小数点	VAD/DTX/CNG
G.723.1 Annex B	5.3/6.3	浮動小数点	浮動小数点演算

● 特許

G.723.1を使用するためには，次の特許所有者よりライセンスを取得する必要があります.

- Universite de Sherbrooke
- France Telecom
- AudioCodes
- DSP Group
- NTT
- Nokia

カナダのシプロ社(http://www.sipro.com/)を通じて契約を行うことができます.

ITU-T G.729 Annex A

CS-ACELP(Conjugate-Structure Algebraic CELP：共役構造代数CELP)と呼ばれます.

G.729 Annex AはG.729の拡張版で，G.729と互換性を保ちながら演算量を半分程度に抑えたバージョンです．若干の品質劣化があるものの演算量が少ないため，G.729よりも頻繁に使用されます.

● 入出力

10msのフレーム単位の処理になります(**図13**，p.99).

デコーダにはフレームエラー情報を入力することができます.

●音声データのフォーマット

16ビットリニアPCM，80サンプル/フレーム

●符号データのフォーマット

80ビット/フレーム

注1：最尤(ゆう)と書く．もっともそれらしいの意.

New Products ― フィリップス，MP3対応1チップICとリファレンスデザインを発表
ロイヤルフィリップスエレクトロニクスは，MP3/AAC/WMAに対応した1チップIC｢SAA7752｣を発売し，同時にICを組み込んだ基板，ソフトウェアスタック，リファレンスデザインの提供を開始する.

〔図12〕
G.723.1 符号データ

( ) 内の数字は各パラメータのビット番号(0 が LSB).

	7 (MSB)	6	5	4	3	2	1	0 (LSB)
[0]	LPC(5)	LPC(4)	LPC(3)	LPC(2)	LPC(1)	LPC(0)	0	0
[1]	LPC(13)	LPC(12)	LPC(11)	LPC(10)	LPC(9)	LPC(8)	LPC(7)	LPC(6)
[2]	LPC(21)	LPC(20)	LPC(19)	LPC(18)	LPC(17)	LPC(16)	LPC(15)	LPC(14)
[3]	ACL0(5)	ACL0(4)	ACL0(3)	ACL0(2)	ACL0(1)	ACL0(0)	LPC(23)	LPC(22)
[4]	ACL2(4)	ACL2(3)	ACL2(2)	ACL2(1)	ACL2(0)	ACL1(1)	ACL1(0)	ACL0(6)
[5]	GAIN0(3)	GAIN0(2)	GAIN0(1)	GAIN0(0)	ACL3(1)	ACL3(0)	ACL2(6)	ACL2(5)
[6]	GAIN0(11)	GAIN0(10)	GAIN0(9)	GAIN0(8)	GAIN0(7)	GAIN0 (6)	GAIN0(5)	GAIN0(4)
[7]	GAIN1(7)	GAIN1(6)	GAIN1(5)	GAIN1(4)	GAIN1(3)	GAIN1(2)	GAIN1(1)	GAIN1(0)
[8]	GAIN2(3)	GAIN2(2)	GAIN2(1)	GAIN2(0)	GAIN1(11)	GAIN1(10)	GAIN1(9)	GAIN1(8)
[9]	GAIN2(11)	GAIN2(10)	GAIN2(9)	GAIN2(8)	GAIN2(7)	GAIN2(6)	GAIN2(5)	GAIN2(4)
[10]	GAIN3(7)	GAIN3(6)	GAIN3(5)	GAIN3(4)	GAIN3(3)	GAIN3(2)	GAIN3(1)	GAIN3(0)
[11]	GRID3(0)	GRID2(0)	GRID1(0)	GRID0(0)	GAIN3(11)	GAIN3(10)	GAIN3(9)	GAIN3(8)
[12]	MPOS(6)	MPOS(5)	MPOS(4)	MPOS(3)	MPOS(2)	MPOS (1)	MPOS(0)	UB
[13]	POS0(1)	POS0(0)	MPOS(12)	MPOS(11)	MPOS(10)	MPOS(9)	MPOS(8)	MPOS(7)
[14]	POS0(9)	POS0(8)	POS0(7)	POS0(6)	POS0(5)	POS0(4)	POS0(3)	POS0(2)
[15]	POS1(1)	POS1(0)	POS0(15)	POS0(14)	POS0(13)	POS0(12)	POS0(11)	POS0(10)
[16]	POS1(9)	POS1(8)	POS1(7)	POS1(6)	POS1(5)	POS1(4)	POS1(3)	POS1(2)
[17]	POS2(3)	POS2(2)	POS2(1)	POS2(0)	POS1(13)	POS1(12)	POS1(11)	POS1(10)
[18]	POS2(11)	POS2(10)	POS2(9)	POS2(8)	POS2(7)	POS2(6)	POS2(5)	POS2(4)
[19]	POS3(3)	POS3(2)	POS3(1)	POS3(0)	POS2(15)	POS2(14)	POS2(13)	POS2(12)
[20]	POS3(11)	POS3(10)	POS3(9)	POS3(8)	POS3(7)	POS3(6)	POS3(5)	POS3(4)
[21]	PSIG0(5)	PSIG0(4)	PSIG0(3)	PSIG0(2)	PSIG0(1)	PSIG0(0)	POS3(13)	POS3(12)
[22]	PSIG2(2)	PSIG2(1)	PSIG2(0)	PSIG1(4)	PSIG1(3)	PSIG1(2)	PSIG1(1)	PSIG1(0)
[23]	PSIG3(4)	PSIG3(3)	PSIG3(2)	PSIG3(1)	PSIG3(0)	PSIG2(5)	PSIG2(4)	PSIG2(3)

(**a**) 6.3kbps の場合

	7 (MSB)	6	5	4	3	2	1	0 (LSB)
[0]	LPC(5)	LPC(4)	LPC(3)	LPC(2)	LPC(1)	LPC(0)	0	1
[1]	LPC(13)	LPC(12)	LPC(11)	LPC(10)	LPC(9)	LPC(8)	LPC(7)	LPC(6)
[2]	LPC(21)	LPC(20)	LPC(19)	LPC(18)	LPC(17)	LPC(16)	LPC(15)	LPC(14)
[3]	ACL0(5)	ACL0(4)	ACL0(3)	ACL0(2)	ACL0(1)	ACL0(0)	LPC(23)	LPC(22)
[4]	ACL2(4)	ACL2(3)	ACL2(2)	ACL2(1)	ACL2(0)	ACL1(1)	ACL1(0)	ACL0(6)
[5]	GAIN0(3)	GAIN0(2)	GAIN0(1)	GAIN0(0)	ACL3(1)	ACL3(0)	ACL2(6)	ACL2(5)
[6]	GAIN0(11)	GAIN0(10)	GAIN0(9)	GAIN0(8)	GAIN0(7)	GAIN0(6)	GAIN0(5)	GAIN0(4)
[7]	GAIN1(7)	GAIN1(6)	GAIN1(5)	GAIN1(4)	GAIN1(3)	GAIN1(2)	GAIN1(1)	GAIN1(0)
[8]	GAIN2(3)	GAIN2(2)	GAIN2(1)	GAIN2(0)	GAIN1(11)	GAIN1(10)	GAIN1(9)	GAIN1(8)
[9]	GAIN2(11)	GAIN2(10)	GAIN2(9)	GAIN2(8)	GAIN2(7)	GAIN2(6)	GAIN2(5)	GAIN2(4)
[10]	GAIN3(7)	GAIN3(6)	GAIN3(5)	GAIN3(4)	GAIN3(3)	GAIN3(2)	GAIN3(1)	GAIN3(0)
[11]	GRID3(0)	GRID2(0)	GRID1(0)	GRID0(0)	GAIN3(11)	GAIN3(10)	GAIN3(9)	GAIN3(8)
[12]	POS0(7)	POS0(6)	POS0(5)	POS0(4)	POS0(3)	POS0(2)	POS0(1)	POS0(0)
[13]	POS1(3)	POS1(2)	POS1(1)	POS1(0)	POS0(11)	POS0(10)	POS0(9)	POS0(8)
[14]	POS1(11)	POS1(10)	POS1(9)	POS1(8)	POS1(7)	POS1(6)	POS1(5)	POS1(4)
[15]	POS2(7)	POS2(6)	POS2(5)	POS2(4)	POS2(3)	POS2(2)	POS2(1)	POS2(0)
[16]	POS3(3)	POS3(2)	POS3(1)	POS3(0)	POS2(11)	POS2(10)	POS2(9)	POS2(8)
[17]	POS3(11)	POS3(10)	POS3(9)	POS3(8)	POS3(7)	POS3(6)	POS3(5)	POS3(4)
[18]	PSIG1(3)	PSIG1(2)	PSIG1(1)	PSIG1(0)	PSIG0(3)	PSIG0(2)	PSIG0(1)	PSIG0(0)
[19]	PSIG3(3)	PSIG3(2)	PSIG3(1)	PSIG3(0)	PSIG2(3)	PSIG2(2)	PSIG2(1)	PSIG2(0)

(**b**) 5.3kbps の場合

- RTP

ペイロードタイプは「18」です.

RTP では, 20ms 以上のパケット送出間隔が推奨されているので, 2 フレーム以上の符号データを一つのパケットにまとめます. **図14** と **表6** に 1 フレーム分の符号データのフォーマットを示します.

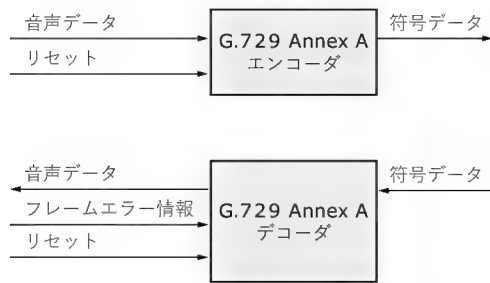
- 入手方法

ITU-T より C ソースコードが入手できます.

- G.729 の拡張機能

表7 のように G.729 には Annex が多数あります.

New Products — Nucleus, ARM10 をサポート
アクセラレイテッド・テクノロジー, メンター・グラフィックス組み込みシステム事業部は, 同社の RTOS「Nucleus」が ARM10 に対応したことを発表した.

〔図13〕G.729 Annex A の入出力

〔図14〕G.729 Annex A 符号データ

( ) 内の数字は各パラメータのビット番号(0 が LSB).

	7 (MSB)	6	5	4	3	2	1	0 (LSB)
[0]	L0(0)	L1(6)	L1(5)	L1(4)	L1(3)	L1(2)	L1(1)	L1(0)
[1]	L2(4)	L2(3)	L2(2)	L2(1)	L2(0)	L3 (4)	L3(3)	L3(2)
[2]	L3(1)	L3(0)	P1(7)	P1(6)	P1(5)	P1(4)	P1 (3)	P1(2)
[3]	P1(1)	P1(0)	P0(0)	C1(12)	C1(11)	C1(10)	C1(9)	C1(8)
[4]	C1(7)	C1(6)	C1(5)	C1(4)	C1(3)	C1(2)	C1(1)	C1(0)
[5]	S1(3)	S1(2)	S1(1)	S1(0)	GA1(2)	GA1(1)	GA1(0)	GB1(3)
[6]	GB1(2)	GB1(1)	GB1(0)	P2(4)	P2(3)	P2(2)	P2(1)	P2(0)
[7]	C2(12)	C2(11)	C2(10)	C2(9)	C2(8)	C2(7)	C2(6)	C2(5)
[8]	C2(4)	C2(3)	C2(2)	C2(1)	C2(0)	S2(3)	S2(2)	S2(1)
[9]	S2(0)	GA2(2)	GA2(1)	GA2(0)	GB2(3)	GB2(2)	GB2(1)	GB2(0)

少々複雑に見えますが，基本的な拡張は次の三つのみで，後はこれらの組み合わせになっています．

- Annex B ： VAD/DTX/CNG
- Annex D ： 6.4kbps
- Annex E ： 11.8kbps

Annex C+，F，G，H，Iはたんに複数のビットレートが使用できるだけではなく，通信中にいつでも(フレーム単位で)ビットレートを切り換えることができます．したがって，ネットワークのトラフィックに合わせた動的なビットレート制御が可能になります．

● 特許

G.729 Annex Aを使用するためには，次の特許所有者よりライセンスを取得する必要があります．

- Universite de Sherbrooke
- France Telecom
- NTT
- Nokia
- NEC

カナダのシプロ社(http://www.sipro.com/)を通じて契約を行うことができます．国内では日本キャステム(株)(http://www.kyastem.co.jp/)が契約のサポートをしています．

〔表6〕G.729 Annex A 符号データのパラメータ

パラメータ	内　容
L0	Switched MA predictor of LSP quantizer
L1	First stage vector of quantizer
L2	Second stage lower vector of LSP quantizer
L3	Second stage higher vector of LSP quantizer
P1	Pitch delay first subframe
P0	Parity bit for pitch delay
C1	Fixed codebook first subframe
S1	Signs of fixed-codebook pulses 1st subframe
GA1	Gain codebook (stage 1) 1st subframe
GB1	Gain codebook (stage 2) 1st subframe
P2	Pitch delay second subframe
C2	Fixed codebook 2nd subframe
S2	Signs of fixed-codebook pulses 2nd subframe
GA2	Gain codebook (stage 1) 2nd subframe
GB2	Gain codebook (stage 2) 2nd subframe

〔表7〕G.729 拡張機能

	ビットレート (kbps)	演算精度	特　徴
G.729	8	固定小数点	
G.729 Annex A	8	固定小数点	低演算量
G.729 Annex B	8 (DTX)	固定小数点	VAD/DTX/CNG
G.729 Annex C	8	浮動小数点	G.729，G.729 Annex Aの浮動小数点演算
G.729 Annex C+	6.4/8/11.8 (DTX)	浮動小数点	G.729 Annex Iの浮動小数点演算
G.729 Annex D	6.4	固定小数点	低ビットレート
G.729 Annex E	11.8	固定小数点	高品質
G.729 Annex F	6.4/8 (DTX)	固定小数点	デュアルレート，VAD/DTX/CNG
G.729 Annex G	8/11.8 (DTX)	固定小数点	デュアルレート，VAD/DTX/CNG
G.729 Annex H	6.4/8/11.8	固定小数点	マルチレート
G.729 Annex I	6.4/8/11.8 (DTX)	固定小数点	マルチレート，VAD/DTX/CNG

おわりに

音声CODECの原理から実際の使い方までを駆け足で説明しましたが，いかがでしたでしょうか？

ビットレート，音質，パケット長，遅延，トラフィックなど相反する条件の中から最適なCODECを選択することは難しい作業です．この記事が少しでもお役に立てば幸いです．

またVoIPにおける音質は，CODECそのものの品質もさることながらCODECとネットワークをつなぐ中間的な部分，たとえばジッタバッファやパケットロスの検出方法などで差が出ます．CODECの特性を理解したうえで，うまく制御することが必要です．

あおき・みのる　日本キャステム(株)

New Products — 小型ヒューマノイドロボット「HOAP-2」発売
富士通オートメーション(株)は，小型ヒューマノイドロボット「HOAP-2」を2003年7月に発売する．身長50cm，体重7kgで，関節自由度は25軸，OSとしてRTLinux，インターフェースとしてUSB 1.0を採用している．オープン価格．

安全なVoIP運用において必要とされる VoIPにおけるセキュリティ

今中宇麻

従来の固定電話では，とくにセキュリティに関して考慮しなくとも安全な運用が可能だった．しかしVoIP機器は，電話機であると同時にインターネットに接続されるIPネットワーク機器でもある．つまり，VoIP機器は他のIPネットワーク機器と同様，さまざまな攻撃を受ける可能性がある．
そこで本章では，VoIPを安全に運用するうえで必要になる知識について解説を行う．
（編集部）

VoIPはその名のとおり，IP上でSIPやH.323やMGCPといったプロトコルを使用することで，Voice（音声）をやりとりするアプリケーションの一種です．HTTPやFTPなどを始めとする他のアプリケーションでは，いまやネットワークのセキュリティ確保は当たり前のことになっていますが，これはVoIPでも例外ではありません．エンタープライズを始め，SOHOや家庭内LANにおけるセキュアなネットワーク構築において，VoIPアプリケーションのセキュリティ確保も非常に重要になります．また，ソフトスイッチもしくはゲートキーパと呼ばれるVoIPトラフィックを正しくルーティングし制御するためのサーバは，その役割上，広範囲なネットワークからアクセスされるように設計されることが多く，それと同時になにかしらのアタックを受ける可能性がある，というセキュリティ上のリスクも高くなってしまいます．

本稿では，VoIPにおけるセキュリティ上の問題やネットワーク構築時に常に問題となるNAT/ファイアウォールの使用にまつわる問題について，エンタープライズ導入における新規VoIPシステムや通信事業者/サービスプロバイダ各社が開始しているIP電話サービスにおいてもっともよく採用されているSIPをメインに紹介していきます．なお，セキュリティにはさまざまなレベルでの話題がありますが，今回はVoIPプロトコルレベルでのセキュリティがメインになります．

〔図1〕DoS/DDoS攻撃

おもなセキュリティ上の問題

VoIPにおいて発生し得るセキュリティ上の問題について，おもなものを以下に記してみました．

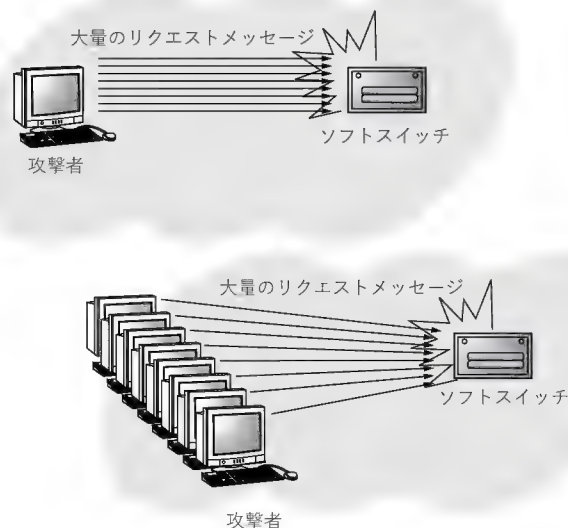
● DoS（Denial of Service）攻撃問題

ある一箇所もしくは複数箇所〔この場合はDDoS（Distributed Denial of Service）になる〕からプロトコルメッセージを一斉に受け取るために，サーバやIP電話端末の処理速度が大幅に低下したり，場合によってはシステム自体が停止してしまうことがあります（**図1**）．なお，以前にいわゆる「ワン切り」業者が大量発呼した（一度に大量の発信を試みた）ことにより，特定地域の電話交換機システムが停止した事件をご記憶のかたもいらっしゃるかもしれません．これも一種のDoS攻撃事件であり，今後VoIPの世界においても同様の事件が発生する可能性は十分にあると思われます．とくにVoIPだと，パソコン1台とネットワークへのアクセスラインが1本あれば簡単に大量発呼ができるなど，特別な設備を準備したり，コストをかけたりすることなく攻撃ができてしまいます．

● 不正プロトコルメッセージ問題

プロトコルの仕様上認められていない文字列や規定されている長さ以上の文字列を含むメッセージを受け取ることにより，サーバやIP電話端末のシステムが停止することがあります．また，場合によっては，バッファオーバフローを引き起こすことで，悪意のある第三者に任意のコードを実行されたり，ルート権限を奪取されたりすることが発生します．

● なりすまし問題

悪意のある第三者が特定のIP電話端末になりすますことで，本来のIP電話端末宛の通話を横取りしてしまうことや，その第

三者が正しく課金されることなく通話を行う(なりすまされたユーザーへ課金されてしまう)ことなどがおきてしまいます(**図2**).

なおSIPでは，IP電話端末のサーバへの登録(自分の電話番号宛の電話が着信できるように，電話番号とIP電話端末のIPアドレスとのバインディング情報をサーバへ登録しにいく)や電話の発信時に，それぞれHTTP Digestによる認証処理を実行することが可能になっているので，なりすまし問題による脅威をある程度取り除くことが可能になっています.

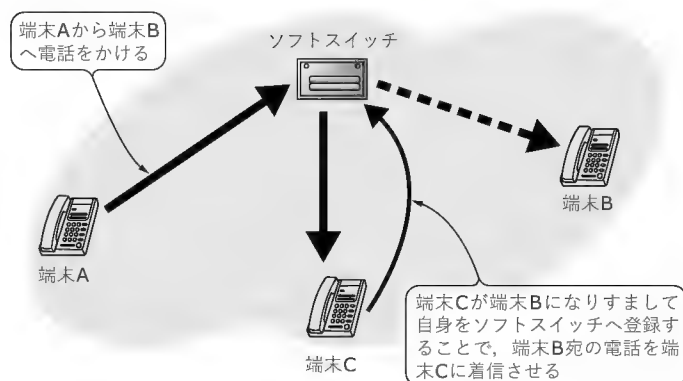
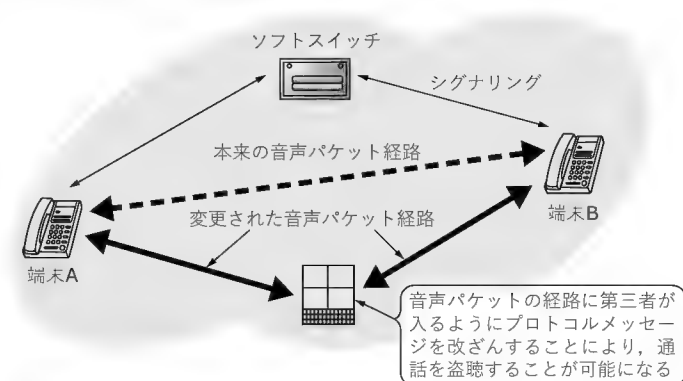
- メッセージの改ざん問題

プロトコルメッセージは，着信端末があるネットワークのサーバまで，場合によっては複数サーバをホップすることがあります.

その経路上にあるサーバのうちのいずれかが，悪意のあるものによって設置されている(もしくは悪意のあるものにより乗っ取られている)場合，プロトコルメッセージの一部が不正に書き換えられる(改ざんされる)可能性があります.

- 音声パケットの盗聴問題

上記メッセージ改ざん問題とも関連がありますが，メッセージ改ざんの際に音声パケットの送信先を悪意のあるものの端末に指定することにより，通話における音声パケットがそこを経由するように設定することができます. これにより，通話は普通にできているが，通話内容が盗聴されているという状況が発生する可能性があります(**図3**).

〔**図2**〕**なりすまし**

〔**図3**〕**音声パケットの盗聴**

SIP実装に関するCERT Advisory

2月末に米国のコンピュータ緊急対応チーム/調整センター(CERT/CC：Computer Emergency Response Team/Coordination Center)より，一部のSIP実装の脆弱性に関するセキュリティ情報(CERT Advisory)が公表されました(**図4**).「CERT Advisory CA-2003-06 Multiple vulnerabilities in implementations of the Session Initiation Protocol(SIP)」(`http://www.cert.org/advisories/CA-2003-06.html`)として公表されたこのアドバイザリによると，一部製品のSIP実装では不正なSIPのプロトコルメッセージに対する対策が十分でないため，サービスの停止やサーバ/端末の乗っ取りが可能になることがあるとのことです.

なお，この問題を回避するための方法として，アドバイザリでは以下の対策が薦められています.

1) ベンダから提供されるパッチを当てる
2) SIP端末やサービスを停止する(ただし，SIPを使用していない場合に限られる)
3) SIPが使用するポート(5060/tcp，5060/udp，5061/tcp)をフィルタリングする(ネットワーク間におけるVoIP通信が必要がない場合)
4) ブロードキャストアドレスに対するSIPリクエストメッセージをブロックする

また，同アドバイザリにはベンダ各社による情報が掲載されているので，どの製品/バージョンがこの脆弱性を有していて，その場合どう対処すればよいかを確認することが可能です.

〔**図4**〕**CERT AdvisoryのWebページ**

ちなみに，この脆弱性を報告した団体であるOulu University Secure Programming Group(OUSPG)のWebサイト(`http://www.ee.oulu.fi/research/ouspg/protos/testing/c07/sip/`)では，実際に本脆弱性を検証するために使用したテストツール(PROTOS Test-Suite: c07-sip)が公開されているの

New Products — 横河電機，ネットワークセキュリティ対策のための日本語脆弱性情報サービスを開始
横河電機(株)は，ネットワークセキュリティ対策のための日本語脆弱性情報サービス「SIOS(サイオス)データベース」を発売した. 価格はプラチナパックが¥1000万/年，ゴールドパックが¥100万/年. 同製品の無償ポータルサイト(https://vdb.sios.biz/)も一般公開される.

で，それを用いてお手持ちのSIP機器や自分でプログラミングしたSIPソフトウェアに対して各種テストを実施してみるのも良いかもしれません．

VoIPネットワークにおけるNAT/ファイアウォールの使用

VoIPネットワークを構築する際に，現在大きな問題となっているものの一つに，「NAT/ファイアウォール問題」といわれるものがあります．この問題は，NATやファイアウォールを用いたネットワークにおいて，VoIP通信を正常に確立させることができないというものです．NATがVoIP通信に与える影響/原因とファイアウォールがVoIP通信に与える影響/原因は，細かいところでそれぞれ異なっています．

なお，本稿ではNATをアドレス変換とポート変換（NAPTやIP Masqueradeとも呼ばれている）の両方の意味を兼ねるものとして使用します．また，NATはたんに割り当てられたIPアドレス数の制限からだけでなく，セキュリティ的側面（NATがセキュアなネットワークを実現するのは必ずしも正しくないのだが，このような効果があることも事実）からも導入を実施しているネットワークがあるので，本章で取り上げます．

〔図5〕NAT使用による問題

〔図6〕ファイアウォール使用による問題

まず，NATを用いることによりVoIP通信を確立させることができない問題についてですが，NATはIPヘッダにあるIPアドレスおよびポート番号のみを変換し，ペイロード（データ）の中身までは関知しないため，SIPのメッセージに記述されているIPアドレスやポート番号などが変換されないまま相手の端末に届いてしまいます．このメッセージを受け取った端末は，メッセージに記載されているIPアドレスおよびポート番号に対してSIPメッセージや音声パケットを送信しようとしますが，NATの背後にあるアドレスはNATを介さないと直接ルーティングできないアドレス（通常はプライベートアドレス）なので，そのメッセージや音声パケットが相手の端末に届かず，VoIP通信を確立させることができません（**図5**）．

次に，ファイアウォールを用いることによりVoIP通信を確立させることができない問題についてですが，これも基本的にはSIPのメッセージに記述されているIPアドレスやポート番号などが変更されないことが原因となります．

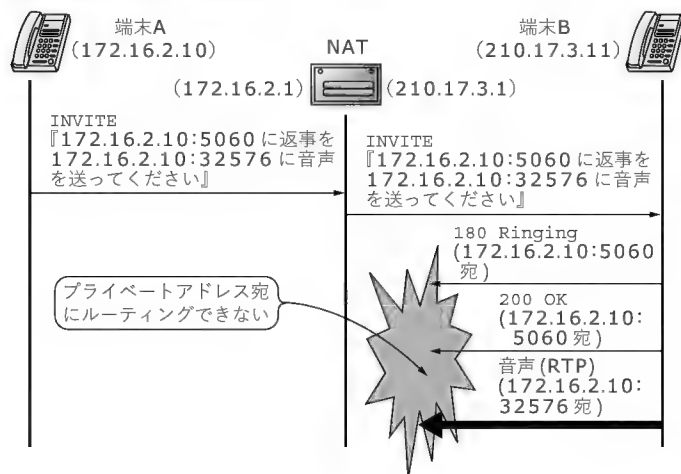
ファイアウォールは，通常，ファイアウォール内部のネットワークから外部のネットワークへのトラフィックは通しますが，外部のネットワークからのトラフィックはあらかじめ許可されたもの，もしくは内部ネットワークから外部ネットワークに向けて送ったメッセージに対するレスポンスメッセージ以外は破棄するように設定されています．そのため，SIPメッセージがファイアウォールを通過する際，ファイアウォールは送信されたメッセージに対するレスポンスメッセージが戻ってくるかもしれないことを学習して動的テーブルを生成しますが，レスポンスメッセージ以外にメディアストリーム（通話音声）のための音声パケットが送信されてくることを認識できないため，その音声パケットをファイアウォールが破棄してしまい，VoIP通信を確立させることができません（**図6**）．

また，音声パケットはUDPを使用しており，通話ごとに使用するポート番号が変更されるため，音声パケット用の許可リストを作成するとなるとほとんどのUDPポートをオープンに設定する必要があり，ファイアウォールはその役割の大部分を果たさなくなってしまいます．

なお，今回の説明ではSIPを使用しましたが，他の主要VoIPプロトコル（H.323やMGCP）もプロトコルメッセージ内部にアドレスやポート番号を記述しているため，同様の事象が発生します．

このように，VoIPにおいてNATおよびファイアウォールを使用するためには，VoIPトラフィックのNAT/ファイアウォールトラバーサル（NAT/ファイアウォール越え）のためのソリューションが必要になってしまいます．

そこで以降では，それらソリューションについて，おもなものを二つ紹介します．

Information ―イノテック，Hier Design社と代理店契約を締結
イノテック（株）は，FPGA用EDAソフトウェアの開発を行うHier Design社と，日本における製品販売とサポートに関して代理店契約を締結した．

UPnP

UPnP(Universal Plug and Play)は，UPnP Forum(http://www.upnp.org/)にて規定された，パソコン/周辺機器や家電機器などの接続性を高め，シームレスなピアツーピア通信を実現させるためのアーキテクチャです．その中でも，VoIP通信においておもに利用されるのは，NAT/ファイアウォールトラバーサルのために必要なInternet Gateway Device(IGD)仕様です．

UPnP IGDを使用することで実現できるものは，おもに，

1) グローバルアドレス(WAN側アドレス)を取得する
2) ポートマッピング(アドレスのバインディング情報)を追加/削除する
3) 現在のポートマッピングを取得する
4) ポートマッピングにリース期間を設定する

となっています．なお，IGDの詳細仕様についてお知りになりたい方は，UPnP Forumから取得できるIGD仕様ドキュメント(http://www.upnp.org/standardizeddcps/igd.asp)や，IGDを提案したインテルのWebサイト(http://www.intel.com/labs/connectivity/upnp/)などを参照いただければ，さまざまな情報を得ることができます．

ちなみに，最近発売されているブロードバンドルータの多くは「UPnP対応」製品として，このUPnP IGD仕様を実装しています．この契機となったのは，マイクロソフトのWindows Messenger/MSN MessengerがUPnP対応になったことであり，ブロードバンドルータによっては「UPnP対応」ではなく「Windows Messenger/MSN Messenger対応」というアピールの仕方を前面に押し出しているものもあるぐらいです．

UPnPを利用することにより，UPnP対応端末はUPnP対応ブロードバンドルータに対してメディアストリーム用のポートマッピング情報を生成させることができます．そして，そのマッピングに用いたグローバルアドレス情報を取得することで，その端末が送信するプロトコルメッセージ内部に記述するアドレスをその取得したアドレスに設定することが可能になり，外部ネットワークからその端末に対してパケットを正常にルーティングさせることができるようになります．これにより，グローバルアドレスが一つしか割り当てられていないが，NATすることによって複数台のパソコンを利用している環境(たとえば家庭内LANなど)においても，音声チャットやビデオチャット機能などを使用することができるようになりました．

なお，Linuxサーバを使用してNATを実現している方のためには，Linux UPnP Internet Gateway Device(http://linux-igd.sourceforge.net/)というオープンソースのソフトウェアが公開されています．本ソフトウェアは同じくオープンソースであるLinux SDK for UPnP Devices(libupnp，http://upnp.sourceforge.net/)を使用してUPnP IGDを実装することで，マイクロソフトのインターネット接続共有サービスの動作をエミュレートしています．UPnPに対応したブロードバンドルータやOS(Microsoft Windows XPおよびMe)を使用していない方は，一度自己責任で試されてみるのも良いかもしれません．

このようにUPnPは，VoIP機器のNAT/ファイアウォールトラバーサルを実現するためのとても良いソリューションのように見えますが，同時に問題点/課題もいくつか存在しています．

まず最初は，対応端末が少ないことです．インストールベースではシェア最大であろうWindows Messenger/MSN Messengerが対応しているのはよいのですが，その他のIP電話端末(ソフトフォン含む)でUPnPに対応しているものはほとんどありません．ただし，これだけUPnP対応のブロードバンドルータが増えてくると，今後はUPnP対応IP電話端末も多数登場すると思われるので，この問題が解消されるのはもしかすると時間の問題かもしれません．

次も上記問題と関連しますが，UPnP機能を利用するにはインターネットゲートウェイ(ルータ)と端末の双方がUPnP仕様を実装する必要があります．つまり，現在出回っているIP電話端末などは，ファームウェアアップグレードもしくは対応端末への買い替えをしなければ使用できません．もちろん，ファームウェアアップグレードができるのであれば良いのですが，中にはそう簡単にはいかないこともあるでしょうし，その際にNAT/ファイアウォールトラバーサルのために端末を買い替えるのもたいへんだと思います(端末数が多い，たとえば企業内ネットワークの端末だと余計にたいへんである)．また，ソフトフォンを自作でプログラミングされる場合もたいへんになるかもしれません．

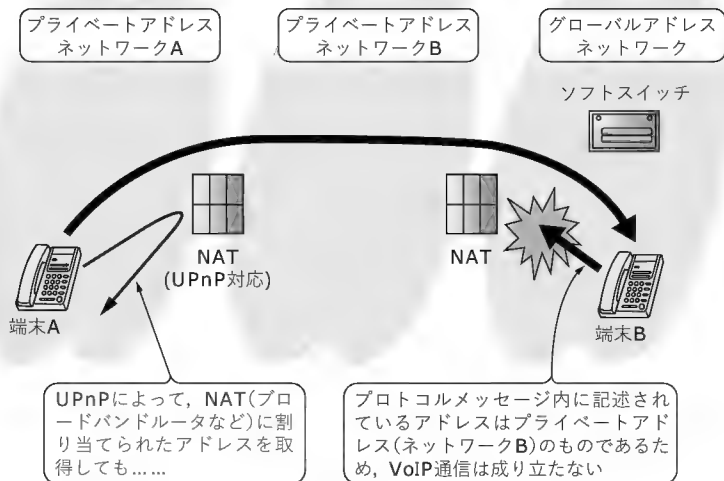
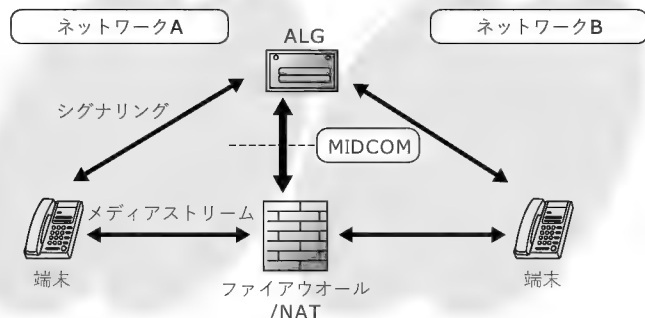
そして，UPnPはルータに割り当てられたアドレスを端末に通知するという仕様上，そのルータにグローバルアドレスが割り当てられていない環境ではVoIP通信を確立させることができません．つまり，NAT/ファイアウォールが多段重ねになっている環境ではUPnP機能を使用することができません(**図7**)．

ただし，UPnPはNAT/ファイアウォールトラバーサルのためだけではなく，家電機器などのネットワーク接続やネットワークへの自動参加を実現させるために使用される仕様でもあることからすると，これからUPnP機能はSOHOや家庭内LANなどの小規模ネットワークにおけるNAT/ファイアウォールトラバーサルソリューションのデファクトスタンダードとして今後より一層普及していくものと，筆者は考えています．

ALG

ALG(Application Layer Gateway)によるソリューションは，ゲートウェイ上にあるアプリケーションによってプロトコルメッセージ内に記述されているアドレスを直接書き換えることにより，VoIPトラフィックのNAT/ファイアウォールトラバーサルを実現します．なお，ALGアプリケーションはNAT/ファイアウォール上に直接実装されている場合もありますし，別のサーバ上にて動作している場合もあります．後者の場合は，ALGア

New Products ―シマンテック，「Symantec AntiVirus for SMTP Gateways 3.1」を発売
(株)シマンテックは，電子メールゲートウェイ向けウィルス対策ソフト「Symantec AntiVirus for SMTP Gateways 3.1」を発売した．今回のバージョンはSPAMメールの判別精度を向上させたことが特徴．

〔図7〕NATの多段構成

〔図8〕MIDCOMプロトコル

プリケーションとNAT/ファイアウォールとの間でバインディング情報の取得やピンホールの設定のためのインターフェースやプロトコルが必要になりますが，現在そのインターフェースやプロトコルは各ベンダ独自のものが使用されています．ただし，この状態だと限られたALGアプリケーションとNAT/ファイアウォールの組み合わせでのみしかシステムを構築することができないため，現在IETFのMIDCOM(Middlebox Communication)ワーキンググループ(http://www.ietf.org/html.charters/midcom-charter.html)では，この用途のための標準プロトコルとして，MIDCOMプロトコルの策定を進めている最中です(**図8**).

ALGを用いることによるメリットは，IP電話端末側に特別なアプリケーションを実装する必要がないところです．もちろん，プロトコルレベルにおける相互接続性の確保は必須になりますが，同一仕様によって実装された端末であればよいため，UPnP対応端末が少ない現状では，UPnPを用いるのに比べるとネットワーク構築の際に選択することのできる端末の種類が格段に多くなります．

逆にデメリットとしては，アプリケーションレベルでプロトコルメッセージの書き換えを実施するため，新しいVoIPプロトコルが導入されたり使用しているVoIPプロトコルの仕様が変更になるたびに，ALGにもサポートプロトコルの追加や機能の変更などが必要になるということがあります．

UPnPと比べ，ALGを用いたソリューションはネットワークをスケールさせることが比較的容易なため，エンタープライズや通信事業者/サービスプロバイダなどの中/大規模ネットワークにおいて，異なるネットワーク間でVoIPトラフィックを送受する場合に使用されることになると筆者は考えています．

おわりに

VoIPが普及するにつれ，セキュリティ問題はよりクローズアップされるトピックになると思われます．現在の電話網は，独自システム/ネットワークを用いており比較的セキュアなものなので，ユーザーがセキュリティについて意識することがそれほどありませんでした．しかしVoIPでは他のデータ通信と同様セキュリティ上の問題に対して認識し，素早くかつ確実に対処することが求められるようになります．

今後増えるかもしれないVoIPに関連したセキュリティインシデントを未然に防ぐためにも，とくにVoIPを利用/導入しているシステム/ネットワーク管理者には，VoIPも十分考慮に入れた状態でセキュアなネットワークの構築および運用を実施することが必要となってくるのではないでしょうか．

いまなか・たかお　NTTコムウェア(株)ビジネスイノベーション本部

Information — Embedded Technology 2003 開催
(社)日本システムハウス協会は，組込み総合技術展「Embedded Technology 2003」を11月12日(水)～14日(金)の日程で開催することを発表した．会場はパシフィコ横浜．

Column

既存LANへVoIPを導入するときの確認ポイント

前村光俊

VoIP導入の形態

VoIPは，音声情報をIPパケットに乗せて運ぶ技術です．VoIPの導入が増えてきましたが，データと音声を同一のLANおよびIP網で通信するため，音声品質に注意する必要があります．VoIPでおもに音声品質に影響を与える要因としては，IP電話機などでの圧縮/伸張による遅延，IP網での遅延や揺らぎがあります(**図A**)．

本社-支店間の通話へVoIPを導入する場合，IP網での遅延や揺らぎの影響を少なくするためには，本社-支店間のネットワークの設計が重要になります．とくに，リアルタイムを要求されないデータ通信で使用していたVoIP導入前の既設網を流用する場合は，ネットワークの特性を把握し，必要に応じて再設計/構築する必要があります．

本稿では，既設社内LAN網にVoIPを導入する際の確認ポイントについて説明します．

既設LAN網のチェックポイント

多くの企業でVoIPを導入するのは，ビル移転，事務所移転などに合わせてすべての社内LANの構築を行う場合と，PBXの設備更改に合わせて既設LANを流用する場合との大きくわけて2パターンが考えられます．新規に社内LANを構築する場合は，レイアウトやデータ通信と音声通信を考慮した最適なネットワーク設計を実施されることと思われます．それでは，既設LANを流用する場合を考えてみます．

● LAN配線の確認

多くの企業ではPBXを導入しており，データ通信と音声通信の2系統のケーブルがくもの巣のように各フロアに敷設されていることが多いようです(**図B**)．ケーブル敷設の初期は，LAN構成図などで管理されていても，長年経つと担当内の席替え，管理者の異動などで初期のLAN構成図と異なって思わぬところでHUBが接続されてLANが分岐/延長されていたり，管理されていない端末が接続されていることがあります．

データ通信では，多少の端末(PC端末など)が増えて衝突や輻輳によるデータのやり取りに多少時間がかかるようになっても気にかける人は少ないでしょう．ですが，電話で時間がかかる(遅延する)と気にかける人は多く，遅延が大きいと通話ができないことになります．つまり，VoIPを導入すると，既設LANにもリアルタイムな通信が求められることになります．遅延発生の要因をなるべく取り除くために，不要な端末やHUBはLANから撤去してLAN構成を把握することが，IP端末を接続した後のトラブル発生時に問題箇所を切り分けるのに有効になります(**図C**)．

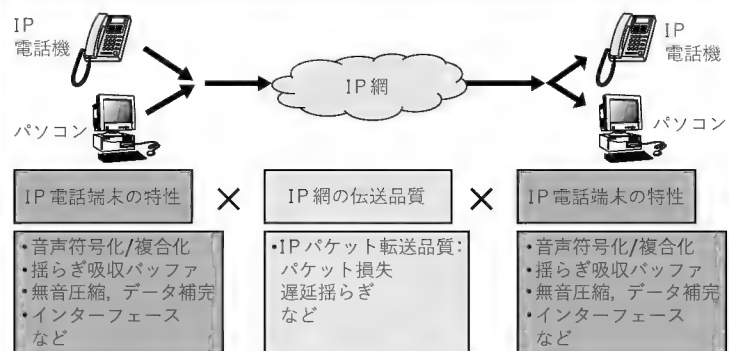
● 空きIPアドレスが必要

IP電話機は，それぞれ個別にIPアドレスが必要になります．また，一般に市販されているアナログ電話機を使用する場合に接続するIAD(Integrated Access Device)についても同様です．社内LANを新たに敷設する場合は，各フロアのPC端末とIP電話機の数量からIPアドレス領域を考慮した設計/構築が可能です．既設の社内LANでは，PC端末の配置や数量を考慮してIPアドレスを確保して運用されているため，設置するIP電話機の数量によって空きIPアドレスに不足がないか確認する必要があります．

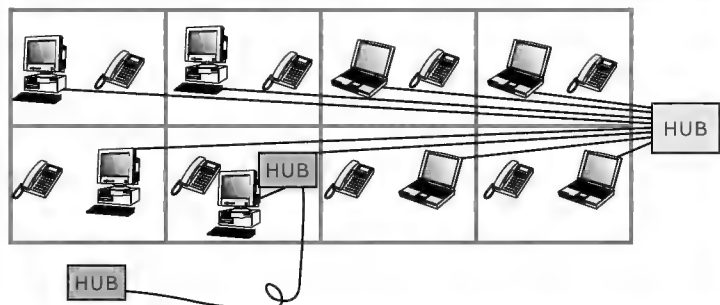
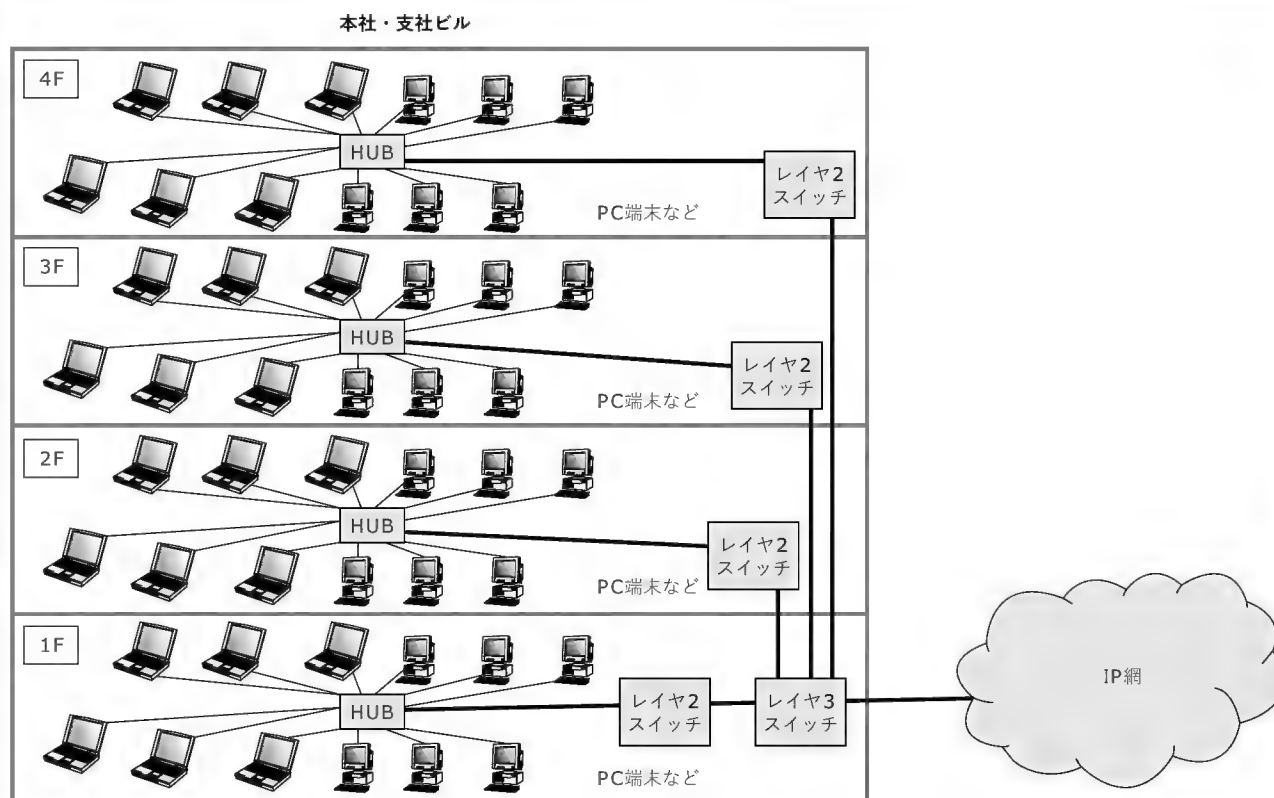
空きIPアドレスが不足する場合には，セグメント分割やNATを使用したIPアドレス体系の見直し，またはDHCPを使用した空きIPアドレスの有効活用が必要となります．新たにIPアドレス体系を見直すと，PC端末などを多く所持するフロアでは，PC端末ごとのIPアドレス変更などの煩雑な作業も発生します．煩雑な作業を考えると，LANをIP電話機用とPC端末用に分割して設計/敷設したほうが簡単かもしれません．

● IP端末の給電を考える

一般の固定電話では，NTTの回線交換機またはPBXの回線から給電されていますが，現在多くのIP電話機は，LANケーブルま

〔図A〕IP端末～IP網～IP端末

出典：「IPネットワーク技術に関する研究会」(第5回)，資料5－4「IPネットワーク技術に関する研究会」報告書(案)より抜粋

〔図B〕フロアのケーブル構成

Information — モンタビスタ，組み込みLinuxでリコーのフォントを採用
モンタビスタソフトウエアジャパン(株)は，(株)リコーのフォントをMontaVista Linuxの推奨フォントと位置づけ，今後組み込みLinuxソリューションの販売/マーケティングに関する販売協力をしていくことを発表した．

〔図C〕ビル内のLAN構成

たは電源ケーブル(ACアダプタ)を使用して電源を供給する仕様になっています.

LANケーブルから電源を供給する場合は，IP電話機まわりにACアダプタのケーブルがなくスッキリと配置できますが，レイヤ2スイッチへの給電機能が必要になります．新規にLAN構築を行う場合には，給電機能をもっているレイヤ2スイッチを選択すると良いのですが，既設LANでLANケーブルからの給電を行おうとするときは，レイヤ2スイッチに給電機能が備わっていないと，レイヤ2スイッチの交換(購入)が必要となります.

ACアダプタを使用する場合は，当然のことながら，設置するIP電話機に応じて電源タップが必要となります．IP電話機によっては，ACアダプタが意外と大きい(ノートPCのACアダプタ程度の大きさ)ので場所を取ります．また，電源プラグと一体型のACアダプタの場合，大きさによっては，タップへの接続数が制限されることもあります.

フロアがケーブル類で煩雑にならないように，電源ケーブルの敷設にも気をつけると良いでしょう.

● 停電時のための対応

一般の固定電話のように，停電時でも使用する方法を考えてみます．ビル内のルータ類(レイヤ3スイッチなど)には，無停電電源装置(UPS)を設置していると思いますが，IP電話機の電源供給元にもUPSが必要となります．IP電話機には停電対策用に電池などを内蔵していない機種がほとんどだと思われます(少なくとも筆者は停電対策された機種を見たことがない)．したがって，LANケーブルを使用してレイヤ2スイッチから電源供給を受ける場合は，レイヤ2スイッチ(HUB)にUPSを準備することが必要となります．また，ACアダプタを使用する場合は，すべてのIP電話機の電源供給元(電源タップ)をUPSに接続するか，停電時に使用するIP電話機を選択して，UPSを使用するなどの対策が必要になります.

● 必要な帯域を確認する

既設LANを流用する場合，VoIP導入前の使用状況で，衝突/輻輳が発生しておらず十分な空き帯域があるのを確認することが必要です．また，すでに電話が導入されていると思われますが，最繁時の通話トラフィックを調査して，VoIPを導入するときにLANに求められる帯域の予測を立てることが必要になります(音声コーデックの選択によって必要帯域が変わるので，帯域を計算するときに考慮が必要).

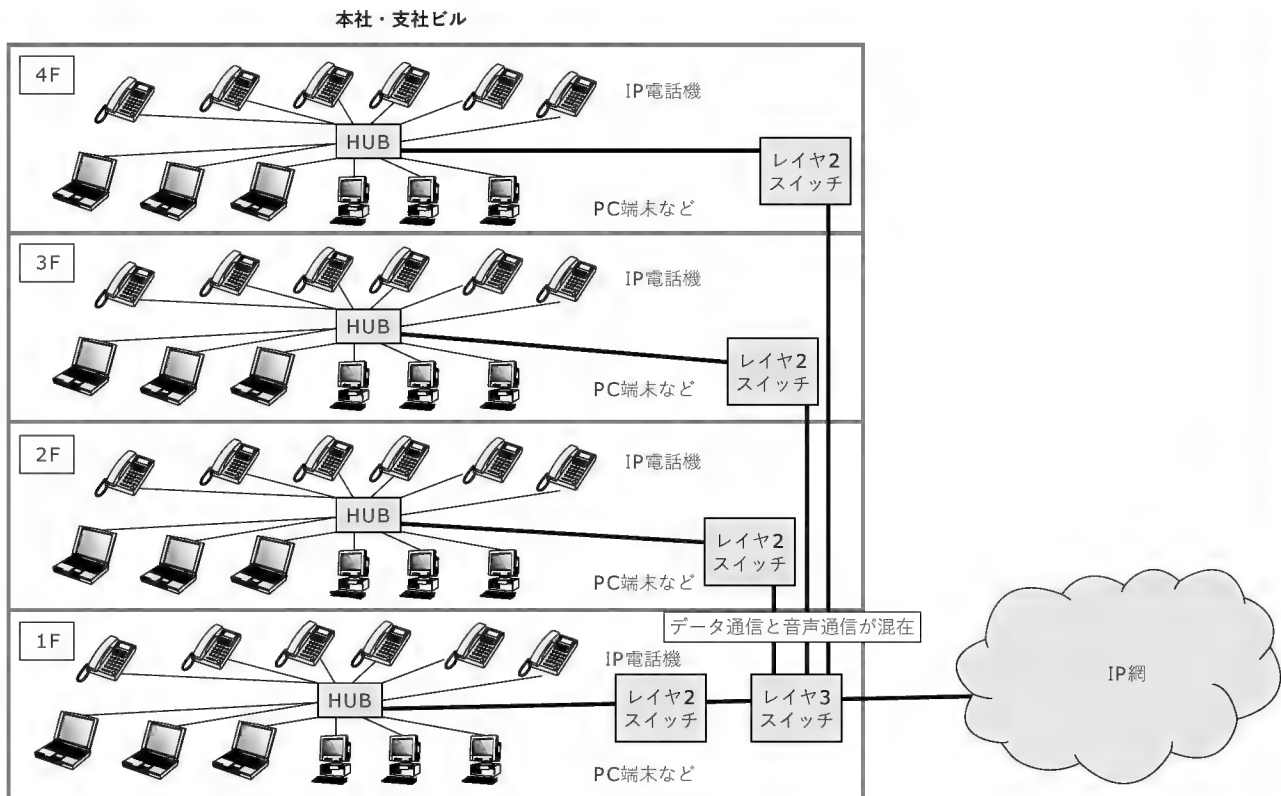
帯域が不足して，音声パケットの損失が多く発生すると，ブツブツと途切れたり，雑音が入った通話になります．また，遅延が発生すると，一昔前の衛星電話のように一言話すたびに，ひと呼吸待って話すような感じになります．VoIP導入後に，このような事象がフロア限定で発生した場合には，LANを調査すると問題解決の糸口になるかもしれません.

● 社内LANの中継装置について

社内LANの中継装置と音声品質との関わりを考えてみます.

IP電話機とPC端末は，レイヤ2スイッチ(HUB含む)を経由して，レイヤ3スイッチ(ルータ)と接続されています(**図D**)．この間は,

New Products ― 光ファイバ線路試験器「OS2090」
(株)システムクラフト(http://www.scinet.co.jp/)は，光ファイバ線路網のシミュレーションが可能な光ファイバ線路試験器「OS2090」を発売した．光ファイバ線路の遅延量と減衰量を個別に設定できることが特徴．価格は¥960,000.

〔図D〕ビル内のLAN構成(データ通信と音声通信が混在する)

データと音声が混在して流れているので,通話品質を維持するためには,レイヤ3スイッチにQoS(Quality of Service)制御機能およびフラグメント機能が必要になります.QoS制御機能は,リアルタイム性が求められる音声パケットを優先的に送信し,遅延を防ぐために使用します.フラグメント機能は,パケット長の長いパケットを細かく分解して,データパケットが送信されるまで音声パケットが待たされたり,廃棄されることがないように送信する機能です.

次のポイントは,レイヤ2スイッチ,レイヤ3スイッチが,全二重通信(Full Duplex)に対応していることです.古いHUBは全二重に対応していないことがあります.PC端末などのデータ通信であれば,受信と送信が片方向ずつの半二重通信(Half Duplex)で,送受信に多少の遅延が発生しても気になりませんが,音声通信を行うには,遅延をできる限り少なくしたリアルタイムで送受信を行わないと,通話が成り立たなくなります.実際にVoIPを導入された方で,通信途切れが発生したケースがありましたが,原因は,レイヤ2スイッチとレイヤ3スイッチ間の通信が半二重通信に設定されていたことでした.

おわりに

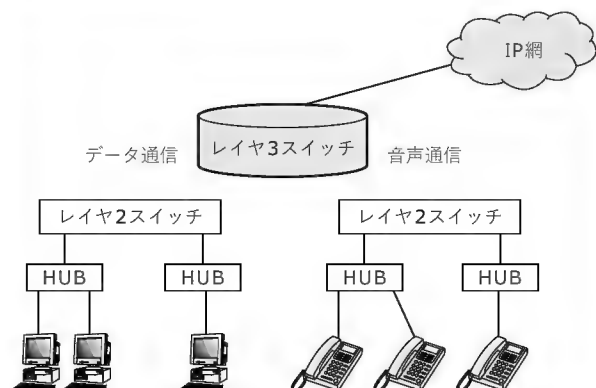
既存LANにVoIPを導入する場合に,必要となる確認ポイントをまとめてみました.

IP電話機での通話品質が,PC端末などのデータ通信の影響を限りなく受けないようにするためには,データ通信用とVoIP用にLANを分割すると良いかもしれません(図E).

この構成は,レイヤ3スイッチのポート単位の優先制御を使用しますので,IP電話機の通信のリアルタイム性を確保できます.VoIP導入後に,音声品質に悩まされている方は,一度検討されてみてはいかがでしょうか?

まえむら・みつとし NTTコムウェア(株)サービス本部

〔図E〕理想のネットワーク

Information ── デジタルテレビ情報化研究会が設立される
シャープ(株),ソニー(株),(株)東芝,(株)日立製作所,松下電器産業(株)は,ディジタルテレビの通信機能を検討するために,「デジタルテレビ情報化研究会」を設立した.

SIP対応ソフトフォンの音質向上技術とビジネスモデル Gphone—— ソフトウェアが電話になる時代

岡崎昌人

ここまでの章では，VoIPの基礎知識とそこで使用される規格について解説を行ってきた．
本章では，VoIPの実装例として，すでに２年以上の稼動実績があり，PC/PDA上で動作するソフトウェアフォン"Gphone"を取り上げ，実用上必要になるジッタの防止や音声バッファリングなどのテクニックについて解説する．
また，VoIPをビジネスとして展開するにあたり必要となる，VoIPのビジネスモデルに関しても取り上げる．（編集部）

2003年現在，さまざまなIP電話が登場してきています．IP電話の物理的形態としては，普通の電話機にIP対応のアダプタを内蔵した形式のものが主流です．これらハードウェアでIP電話を実現しているものに対して，ソフトウェアでIP電話を実現しているものもあり，これを一般に「ソフトフォン」と呼んでいます．

ソフトフォンは，さまざまな可能性を秘めています．その中で特筆すべき点として，ソフトフォンはIP電話の特性をほとんどすべて継承できるということがあげられます．よくIP電話のすばらしさを説明するのに用いられる"コスト削減"というメリットがありますが，これももちろん当てはまります．現在，すでに大企業のインフラはIPベースへ移行しています．これを利用すれば，会社に多くの支店，とくに海外に支店がある場合は，システムを変えることにより経費の削減が実行できるのです．

しかし，これは中～長期的に見たときの話です．なぜなら従来のIP電話では，初期費用が相当額発生するためです．しかしソフトフォンは，ハードウェアを必要としないので格安に導入することができます．ソフトウェアをインストールするだけで，そのパソコンはIP電話に変身するのです．

本章では例としてWindows上で動作する"Gphone"を取り上げ，ソフトフォンの現状について解説を行います．

パソコンが電話になるGphone

現在，Windows向けのPC Gphone 2.0（**図1**）とPocket PC 2002デバイス上で動作するPocket Gphone 2.0（**図2**）があります．前バージョンとなるGphone 1.Xは，フリーソフトウェアとして2001年に発表し，２年が経過しました．今ではGphone以外にも各サードパーティからリリースされています．

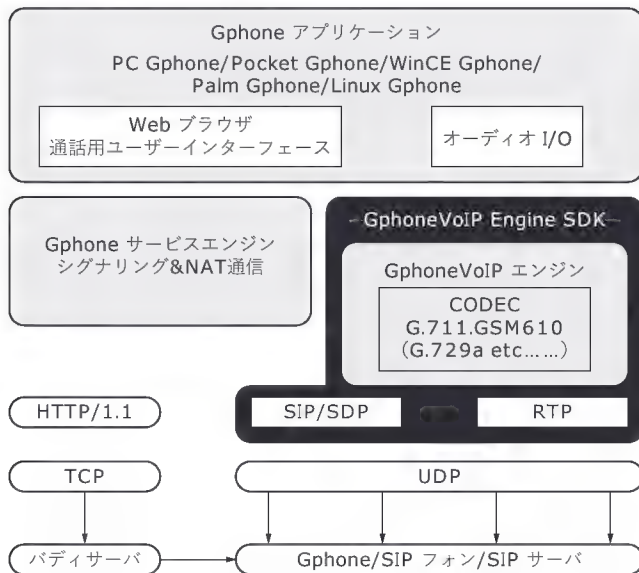
Gphoneにはモバイル対応のソフトフォンエンジン「Gphone VoIPエンジン」を搭載しており，プログラムが非常に軽量で，なおかつ音声品質に優れています．本製品は当初から小型デバイスをターゲットにして設計を行い，開発したものです．

昔話になりますが，２年前のことです．筆者はゲームボーイカラーのような風貌をしたPDA"パームサイズPC"をもっていました．そのPDAのおもな使い道はスケジューラと電話帳で，これだけでも当時は満足していました．ある日，PDAの中に入っている電話番号を調べるために，PDAの電源を入れ，左手には携帯電話をもっていました．PDAに映し出された電話番号を見て，それを携帯電話に入力−−「この二つの機械が一つにならないものか？」と思ったのが，PDAで電話ができるPocket

〔図1〕PC Gphoneの画面

〔図2〕Pocket Gphoneを起動した画面

〔図3〕Gphoneの構成図

〔図4〕バディリスト

Gphone開発のきっかけでした．

“パームサイズPC”と呼ばれる機種では，現在のPocket PC2002とは異なり，標準のオーディオデバイスが半二重で，“音声の再生中は，マイクの入力ができない”という制限がありました．そのため，最初にできあがったアプリケーションは，喋るときにはボタンを押すトランシーバのようなものでした．筆者は，その試作バージョンの入っているPDAに，当時発売されたばかりのPHSカードを挿し，自慢げに持ち歩き通話テストを行いました．さすがに，まわりの人からは変な目で見られました．ゲームボーイを片手に独り言を言っている奴がいるとでも思われたのでしょう．

Gphoneの特徴

Gphoneは**図3**の構成をもっています．TCP/IPスタックと，オーディオデバイス，Webブラウザ以外はすべて独自技術でVoIP機能を実装しています．アプリケーションとしての機能は，音声通話以外にも，テキストチャット，ファイル転送機能，テロップ形式のインフォメーション機能，ホワイトボード機能（次期バージョンより実装）をもっています．これらの機能により，一般電話ではできなかった新しいコミュニケーションの手段をもてるようになります．

それでは，音声通話の機能に着目し，特徴を説明します．

1）小さなフットプリント

Gphoneの音声処理エンジンは50Kバイトに満たない非常に小さな容量に納められています．このためPC以外の小型デバイス（PDAや組み込み用途など）にも容易に搭載が可能です．また，SIPスタックも含んでいます．

2）省スペック環境下での快適動作

Gphoneの音声処理エンジンは，CPU負荷の少ない独自の並列処理技術/バッファリング技術を採用しています．そのためPDAのような処理能力が低いデバイス上でも快適に動作します．

これにより，リソースを大きく消費しないため，ほかの作業を行いながらGphoneを利用するケースなどのバックグラウンド処理でも威力を発揮します．

3）遅延の少ない高品質な音声通話が可能

ソフトフォンでは，音声を送信する際の録音，圧縮，音声を受信する際の展開，再生のためにインターネット網を通すと数秒の遅延が生じます．Gphoneの音声処理エンジンでは，送受信時に処理する1パケットごとの区切り値を最適化し，タスク処理部および音声変換部のエンジンに最適化処理を行うとともに，無駄なパケット送信の検知を独自のテクノロジで行っており，さらに動作が軽いのが特徴です．

4）SIPに対応

SIP（Session Initiation Protocol）に対応しており，ほかのSIPに準拠したSIPフォンとの相互接続が可能です．

5）NAT越えに対応

UPnP（Universal Plug and Play）を使ったNAT越え（Network Address Translation）だけでなく，ファイアウォールなどのセキュリティに守られたLANの中でも，内線電話として用いることができるシステムになっています．

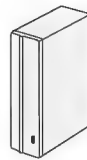
6）バディリストを装備

バディリスト（**図4**）は，接続されたバディサーバからデータを取得します．バディサーバとは，Gphoneを利用するユーザーのアカウント情報，接続情報などを一括管理し，ユーザーへの利便性を提供するサーバです．ユーザーは，バディサーバを利用することにより，IPアドレスやポート番号などを意識することなく容易に特定のユーザーと接続することが可能です．また，サーバ上でユーザー情報を管理しているため，複数のデバイス上でGphoneを利用しているユーザーなどでも，情報の同期は必要ありません．

Information — STマイクロ，スマートカード分野での買収を発表
STマイクロエレクトロニクスは，スマートカードソフトウェアを開発するプロトン・ワールド・インターナショナル（PWI）の買収を発表した．PWIはセキュリティや決済，本人確認のためのスマートカードシステムに特化した企業．

〔図5〕SIPのシーケンス

sip:1082@vmserver
192.168.0.82

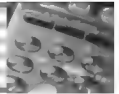
sip:vmserver
192.168.0.93

```
REGISTER sip:vmserver SIP/2.0
Via:SIP/2.0/UDP 192.168.0.82
CSeq:2442 REGISTER
To:<sip:1082@vmserver>
Expires:600
From:<sip:1082@vmserver>
Call-ID:998102360@192.168.0.82
Content-Length:0
User-Agent:Gphone
Contact:" yasunori nakota "<sip:yasun@192.168.0.82;transport=udp>
```

```
SIP/2.0 100 Trying
Via:SIP/2.0/UDP 192.168.0.82:5060
To:<sip:1082@vmserver>
From:<sip:1082@vmserver>
Call-ID:998102360@192.168.0.82
CSeq:2442 REGISTER
Content-Length:0
```

```
SIP/2.0 200 OK
Via:SIP/2.0/UDP 192.168.0.82:5060
To:<sip:1082@vmserver>;tag=d8fb43d9
From:<sip:1082@vmserver>
Call-ID:998102360@192.168.0.82
CSeq:2442 REGISTER
Expires:600
Contact:
<sip:yasun@192.168.0.82;transport=udp>
Content-Length:0
```

```
INVITE sip:1084@vmerver SIP/2.0
Via:SIP/2.0/UDP 192.168.0.82
CSeq:3128 INVITE
To:<sip:1084@vmserver>
Content-Type:application/sdp
From:<sip:1082@vmserver>;tag=16E5
Call-ID:1116229620@192.168.0.82
Subject:sip:1082@vmserver
Content-Length:148
User-Agent:Gphone
Contact:" yasunori nakota "<sip:yasun@192.168.0.82;transport=udp>

v=0
o=username 0 0 IN IP4 192.168.0.82
s=Welcome
c=IN IP4 192.168.0.82
t=0 0
m=audio 33196 RTP/AVP 0
a=rtpmap:0 PCMU/8000
a=ptime:20
```

```
SIP/2.0 100 Trying
Via:SIP/2.0/UDP 192.168.0.82:5060
To:<sip:1084@vmserver>
From:<sip:1082@vmserver>;tag=16E5
Call-ID:1116229620@192.168.0.82
CSeq:3128 INVITE
Content-Length:0
```

ソフトウェアが電話になる理由

次に，どのようにすれば電話と同等の機能をもたせることができるか，そのカラクリを解説していきましょう．

● インターネットを利用したデータ送受信

インターネットにはさまざまなデータが流れています．現在おもなデータとして，TCP/IPによるWebページのデータや，メールなどのデータがあります．IP電話もこれらと同じように音声データをディジタル化してインターネット上に流します．

一般電話の場合は電話番号により，通話相手を特定していますが，IP電話ではIPアドレスによって相手を特定します．IPアドレスをそのまま表記すると数字の羅列でわかりにくいため，ドメイン名とIPアドレスを対応させ，ドメイン名を使って相手に電話することを可能にしています．

実際に相手のIPアドレスがわかり，その相手に通話するにはどういう流れになるかを簡単に説明します．

● シグナリングの手法

まず，相手の端末(IP電話)のベルを鳴らさなければなりません．それには，通話相手先とプロトコルを合致させておく必要があります．もし，別々のプロトコルであれば，相手のベルを鳴らすことすらできません(ここではSIPというプロトコルを使った例として説明を続ける．**図5**)．プロトコルが同じであれば，「ベルを鳴らせ」という命令が到達し，相手側の端末のベルが鳴り，ベルを鳴らしたという知らせが相手側の端末から届きます．

次に相手側が受話器を取ると，先ほどと同様に，相手側の端末から通話開始の情報が届き，実際の音声データの送受信が始まります．この音声データは1秒間に50回分のデータパケットとして分けられリアルタイムに送信されます(つまり1パケットあたり20msの長さの音声データになる)．

そして，会話が終わり相手が受話器を置くと，受話器を置いた側の端末から通信終了を意味する知らせが届き，一連の通話が完了します．

● オーディオ入出力デバイス

では，どのようにして音声をディジタル化して送信するのかを説明しましょう．まず，パソ

New Products ― インフォシーク，インテルなどの協力により検索エンジンを開発
(株)インフォシークは，インテル(株)，オムロンソフトウェア(株)，(株)データ変換研究所の協力により，ASP用検索エンジンを開発した．商用UNIXサーバをIAサーバに置き換えることにより，価格を従来の1/10に，検索速度を5倍にしたことが特徴．

〔図6〕ウェーブフォーマット構造体

```
WAVEFORMATEX              wfx;
//ウェーブフォームオーディオデータのフォーマットを定義するの構造体.
//この構造体には，ウェーブフォームオーディオデータ形式すべてに共通のフォーマット情報が含まれる

wfx.wFormatTag            =WAVE_FORMAT_PCM;//圧縮されていないオリジナルのサンプリングデータをあつかう
wfx.nChannels             =1;             //モノラル1チャネル
wfx.wBitsPerSample        =16;            //分解能16ビット
wfx.nSamplesPerSec        =8000;          //サンプリングレート．1秒あたりのサンプル数(Hz)で表す
wfx.nAvgBytesPerSec       =wfx.nSamplesPerSec * (wfx.wBitsPerSample/8);
wfx.nBlockAlign           =(wfx.wBitsPerSample/8) * wfx.nChannels;
```

コンにマイクロフォン入力と，スピーカ(ヘッドフォン)出力が装備されている必要があります．また，音声入力中に，音声出力が同時にできる必要があります．このFull duplex機能がなければ，トランシーバのように話している時間は相手の声を聞くことができないことになってしまいます．

● 音声のサンプリング

音声をデータとして扱うには，次のような方法を用いるのが一般的です．モノラル1チャネルの8000Hzで音声サンプリングし，分解能は16ビットで行います(**図6**)．この条件により，1秒間，音声のサンプリングを行うと，8000×(16÷2)＝16000バイトのデータが取得できます．

● エンコードと音声データのパケット化

このデータを元に，音声をエンコード&パケット化して送信するデータの準備が完了します．ここでいうエンコードとは，音声の圧縮を意味します．ただし，サンプリングしただけのオリジナルデータをそのまま送信すると128kbpsと大きくなるため，通常はこれを圧縮して送ります．

Gphone VoIPエンジンとは

VoIPエンジンは，“プロトコルとCODECが同じであれば通話品質は変わらない”と思われがちですが，じつはそうではありません．

VoIPのとても重要なポイントとして，遅延と音飛びの制御があります．ここでいう遅延とは，話者Aが，相手Bに音声が到達するまでの時間をいいます．総務省は150ms以内の遅延を“携帯電話並”としているのですが，ソフトフォンでこの基準をクリアするにはかなりの工夫が必要です．

遅延と一言にいっても，さまざまな箇所で遅延が発生します．ネットワークの遅延，ルータを経由するときに発生する遅延，バッファリング遅延，TCP/IPスタックの遅延，CODECによる遅延，タスク処理遅延などです．この中でVoIPエンジンと関連のある遅延は，とくにバッファリング遅延とタスク処理遅延です．

バッファリングはどうして必要なのか，バッファなのだから遅延を少なくするには単純にサイズを小さくすればいいのでは？と思うかもしれませんが，これは音飛びの発生に大きく関わってくるのです．

バッファリング技術としてジッタ調整バッファリングと呼ばれているものがあります．ジッタ調整バッファというものは，絶え間なく届く音声パケットに対し，インターネット経由などで遅延が発生しても，バッファリングによってプールされているパケットを再生し，それが空になるまでの間は音飛びが発生しないようにする技術です．この技術はおもに動画のストリーミングなどでよく見かけます．

しかし，バッファが不十分であれば，再生する音声データがなく，次の音声データパケットが届くまで音が再生されません．この音のとぎれが音飛びとして耳障りに聞こえるのです．逆にジッタバッファを多く取ると，バッファとしてプールしている時間がそのまま遅延時間となります．すなわち，ジッタバッファの制御は音飛びと遅延に大きく関わっていることを意味し，バッファ量の大きさは遅延時間と比例し，音飛びの発生数と反比例しているのです(**図7**)．

図7の左上にはインターネット網から届くパケット(P)を表しています．大きさはさまざまであり，なおかつ，届く間隔は必ずしも一定ではありません．ジッタバッファリングでは，音声データが一定時間分たまるまで，デコードを繰り返し，そのデータをプールします．**図7**では七つのパケットをデコードした時点で，オーディオデバイスに出力しています．

Gphone VoIPエンジンでは，独自のバッファリング技術を用い，バッファリング遅延とタスク処理遅延を最大50ms以内に納めながら，音飛びの発生数率を最小限に抑えることができました．それは一般的な“ジッタ調整バッファリング”に対し，Gphone VoIPエンジンでは，“リアルタイムジッタバッファリング”という技術を取り入れたからです．

リアルタイムジッタバッファリング

リアルタイムジッタバッファリングは，サンプリングを行って取得した音声データを解析し，音声の空白部分をより自然に圧縮します．そして，圧縮前後の間隔における処理時間をうまく使い，ジッタバッファ調整を行うというものです．また，それと同時に，“到着するパケットの時間間隔”を監視し，最適なジッタバッファ数に調整するダイナミックジッタバッファリング機能も兼ねそなえているのです．

New Products ―アジレントのIP電話の音声品質測定のトータルサービスソリューション
アジレント・テクノロジー(株)は，VoIPの音声品質評価に関して，「VoIPネットワーク音声品質測定トータル・ソリューション・サービス」を発売した．

〔図7〕一般的なジッタバッファリング

ジッタバッファリングとデコード

1）ネットワークから入ってきたパケットをバッファに蓄積する（ジッタ調整バッファリング）
2）一定のバッファがたまればそれをデコードする
3）オーディオ出力用のバッファに転送し，必要があればバッファの並びを調整する
4）転送したバッファを用い，音声の再生を行う
5）次のオーディオ出力用のバッファを用意し，1）を再び繰り返す

デメリット：
無音/有音にかかわらず，ネットワークから入ってきたパケットを一定量待つ必要がある．
それは系数 n によって変化し，n が大きくなるにつれ遅延が増大し，n が少なすぎると音飛びの原因となる

1度にデコードすることにより，瞬間的なCPUの利用率が上がり，エンコードと重なったとき，送信時のCPU処理に悪影響を及ぼす

この手法によりVoIPアプリケーションは，さまざまな恩恵を受けることになります．まず，無音部分を圧縮することにより，室内利用での通常の話し方であれば，64kbpsのものが平均20kbpsにまで圧縮できます．また，CODECと組み合わせることで平均5kbpsにまで圧縮できることもわかっています．その上，CPUの占有を極力さけることで，ほかのアプリケーションと併用して使っても問題が出ません．

〔図8〕CPU使用率のグラフ

図8を見てください．これはWindows2000上で各種ソフトフォンを起動し実際に通話したときのCPU利用率です．三つとも同じ動作環境で計測しました．いちばん上のグラフはPC Gphone Ver.2.00で，下の二つは現在Web上で公開されている某有名ソフトフォンです．

このグラフの縦軸はCPUを占有したパーセンテージで，横軸は時間を表しています．初め左に大きな山ができていますが，これはソフトの起動時にできたもので，実際の通話は次の目盛りから始めました．見ていただいてもわかるようにGphoneは小さな山しかありません．これはCPUを占有しない軽い動作といえるでしょう．

Gphoneサービスエンジン

Gphoneで搭載しているエンジンはVoIPエンジンだけではなく，Gphoneサービスエンジンと呼んでいるもう一つのエンジンがあります．このエンジンは，サービスだけに特化したものです．

Gphoneサービスエンジンがなくても VoIPによる通話はできますが，より快適な使い勝手を実現するためには不可欠です．

たとえば，SIPでは規定されていない，登録メンバのネットワークステータスを調べるプレゼンス機能，ボイスメッセージング

機能，異なったSIPネットワークにも発信することのできるローミング機能，これらが利用できるようになります．

● ボイスメッセージング機能

ソフトフォンは，端末となるPCやPDAの電源が入っていないと着信できないのが欠点ですが，それをカバーするサービスがボイスメッセージング機能です．

ボイスメッセージング機能は，オフラインの相手にメッセージを残すことができ，しかも残されたメッセージはいろいろな場所，いろいろなデバイスから聞くことができます．たとえば，海外の出張に行くことになったとします．そこで，フライト中に日本から保存された携帯電話への留守番メッセージを確認する必要があったとしても，それを簡単に聞くことはできません．

しかし，ボイスメッセージング機能なら当然聞くことが可能です．たとえば，空港にあるインターネットサービスでWebページを開き，自分のGナンバ(後述)とパスワードを使ってサインインするだけで，メッセージの一覧が表示され，聞きたい用件をクリックすれば音声データがストリーミング再生されます．もしくはメールに転送するという設定にしておけば，自分のメールアカウントに留守中に保存された音声データが添付されて届くのです．

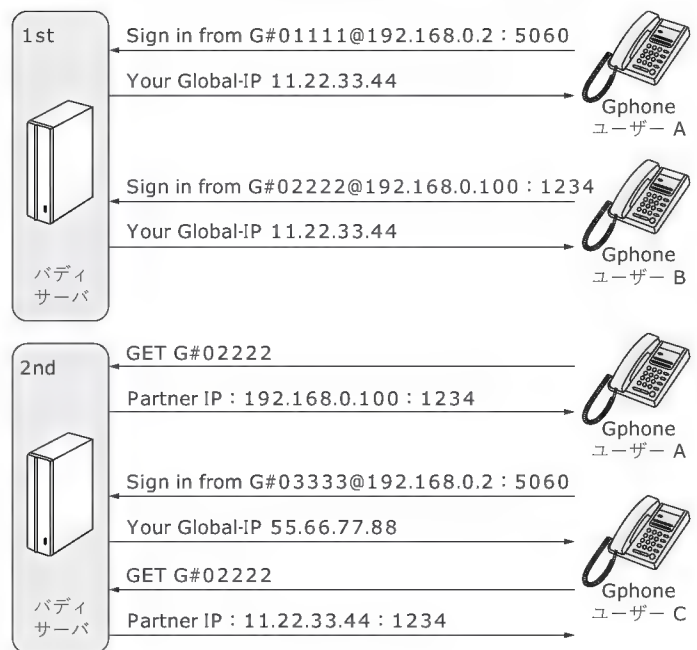
● プレゼンス機能

通話相手の特定はIPアドレスが基本となります．しかし，一般のコンシューマにはIPアドレスはなじみがありません．そこで現在では，各社さまざまな手法を用いてIPアドレスを違う文字列にマッピングしています．GphoneではGphoneサービスエンジンにより，IPアドレスをGナンバというものにマッピングしています．GナンバはIPアドレスだけでなく，あらかじめ登録されていたユーザー情報や，オンライン/オフライン/話し中などのステータスともリンクしており，リアルタイムに更新されます．

これらを管理するサーバをバディサーバと呼んでいます．バディサーバへはHTTP(Hyper Text Transfer Protocol)により通信を行います．この通信は非常に短いセッションで，中のデータ自体も100バイト程度と，サーバにも回線にも負担をかけません．

まず，Gphoneはバディサーバへ認証を行い，通話を待ち受ける自分のネットワークの環境を伝えます．この時点で自分の使っているIPアドレスと使用ポート番号がGナンバにマッピングされます．さらに，ネットワークの環境がリアルタイムに変化しても問題ないように更新を定期的に行います．

ユーザーはバディリストに通話相手のGナンバをセットすることでニックネームが表示され，今後GナンバやIPアドレスを意識することなく，いつでもニックネームを指定するだけで相手のベルを鳴らすことができるのです(**図9**)．

バディサーバは，クライアントとセッションを張ったときのグローバルIPアドレスと，クライアント側で取得したローカルIPアドレスの二つを管理します．これらのIPアドレスは，用途に応じてローカルIPアドレスか，グローバルIPアドレスかを判断し，同一LAN内だけでの通話と判断されれば，ローカルIPアドレス間だけのピアツーピア通信になり，セキュアな状態で通話をすることができます．

〔**図9**〕**GナンバとIPアドレスのマッピング**

一方，同一LAN内に相手がいないと判断された場合は，グローバルIPアドレスとポート番号によって，待ち受けているクライアントを特定します．これは，NAT環境下でも，GK(Gate Keeper)などを用いずとも複数のクライアントが同時にLANの外から着信を受けられるシステムになっているのです．

● ローミング機能

IP電話のISP間相互通話が話題になっています．もともと同じプロトコルを使った通話であるのなら，相互通話は技術的にそれほど難しい問題ではありません．難しいのはサービスの提供をどう行うかでしょう．

Gphoneサービスの範囲は，Gphone to Gphoneだけの接続ではなく，Gphone to 一般電話，Gphone to SIP Phone(SIP User Agent：ここではGphone以外のSIPフォン)も対応しています(**図10**)．

SIPではセッションの確立を行う前に，SIPフォンは，`REGISTER`メソッドを使い，自分の位置情報を自らの属するドメイン管理するレジストラに登録しています．そしてレジストラは，ユーザー名などの含まれるSIP URIをクライアントにマッピングし，ロケーションサーバに登録します．

Gphoneサービスでは，ここまでの流れをすべてバディサーバが行っています．バディサーバに対し認証をすませると，バディサーバはその瞬間，Marshal ServerにSIPの`REGISTER`メソッドを使った登録を行うよう，隣接のSIP Proxyに対し呼びかけます．ここでユーザー情報としてGナンバにドメイン名をつ

けた形のものがSIP URIになり，他のSIPネットワークとのローミングが果たせるようになるのです．

このようにGphoneサービスでは，バディサーバだけのシンプルなネットワークシステムでも稼働し，ほかのSIPネットワークとの接続およびその管理では，サービスと密着した構築が可能なようになっています．

VL Inc.のVoIPビジネスモデル

● VL Inc.について

VL Inc.(旧社名： Gtony,Inc.)は，2000年8月に米国シリコンバレーで設立されたモバイル用VoIP製品をメインとしたモバイル機器向けコミュニケーションソフトウェアを研究・開発する企業です(**図11**)．設計当初からVoIPのモバイル化を考慮した軽いエンジンを考慮し，軽量なVoIPエンジンの開発に成功しています．コアとなるSIP対応Gphoneエンジンは非常に小さく，省リソースCPU環境下でも高品質に動作することが可能です．

〔図10〕Gphoneを使った相互接続

〔図11〕VL Inc.のWebページ

VL Inc.のミッションは「Gphone Everywhere」の世界を実現することです．そのために「小さくて軽く，あらゆるプラットホーム上でも同品質で動作する次世代VoIPモバイルコミュニケーション技術を世の中に送り出すこと」を目標として掲げています．そして，通信可能なすべての機器がコミュニケーションのできる日を目指し，今後のコミュニケーション市場に貢献したいと考えております．

● クライアントユーザーの現状

旧モデルのGphoneバージョン1.60は約2年間にわたり無料で配布されてきました．また，ユーザー数は80か国を超える国々におよそ30万人を数えており，日本国内ではそのうち3割強のユーザーが存在します．

2003年春にはバージョン2.0版の配布を予定しており，ここでは初めてバディサーバを用いたバディサービスを投入する予定となっています．Gphone 2.0ではさまざまな改良を施し，VoIPエンジンの性能向上により音質，遅延がより改善されました．

● 販売モデル

VL Inc.では，大きく分けて五つの販売モデルをもっています．

- PCおよびPDAメーカーへのライセンス販売
- 組み込み用GphoneエンジンおよびSIPスタック類の販売
- 個人ユーザーへのクライアント販売
- 企業ユーザーへのクライアントおよびSIPサーバ販売/ASP事業
- 通信事業者へのクライアントおよびSIPサーバ販売/ASP事業

本拠地である米国以外にも，日本における市場開拓を行っており，NET&COM2003など国内の展示会に出展し，さらなる市場の開拓を行っています(**写真1**)．また，日本国内では今後サービスパートナー企業と共に国内でのサービス活動を行う予定です．

● クライアントライセンス形態

個人ユーザー向けクライアントライセンス形態では，製品そのものではなく，バディサーバへの接続サービスにつき課金を行います．現在では，60日間はフリーでお試しいただくことができ，60日後も継続してバディサービスをご利用いただく際には年間10ドル前後の課金を行っています(日本円での販売も検討中)．しかし，Gphone 1.60と同様，接続時にパートナーへのIPダイレクト接続のみをご利用いただく場合は，無料での利用が可能です．

今後のIP電話の可能性について

日本国内では，「IP電話」そのものの認知度が最近非常に高くなってきています．そして，今後もいわゆる「IP網」を用いたコミュニケーション市場は050の番号付与とともに大きく変化す

るものと考えられます．あらゆる通信事業者がIP電話サービスに参加していることからも，そうならざるを得ない方向で進んでいるのは確かであるといってもいいでしょう．

しかし，VoIPの世界でいえば，今後は「格安通話のほかに何ができるか？」が問われる時代になります．

IP網を使用するわけですから，当然今までと同じような形態のままでは「通話が安くなっただけ」以外に何のメリットも見出すことはできません．今では，インターネットへ接続する通信費用は限りなく安くなり，同コストのまま世界中とつながる時代はすでに構築されています．

VoIPの利点として当然ですが，ディジタルデータを用いることができます．よって，“いかに音声パケット以外のデータを用いた有効なサービスを提供できるのか”という部分が今後の課題となり得ます．

また「音声とデータの同時活用」は，VoIPの最大のメリットであり，こういったサービスが一つのアプリケーション上から可能になってからでこそ，VoIPの最大の効果を発揮するものでしょう．

では，“どのようなデータ利用形態がもっとも必要とされるのか”ですが，これは個人そして企業によって，さまざまな利用形態があるために一概にはいえませんが，たとえばファイル交換やテキストチャットなどはその応用例と考えられます．

一般電話が近年すべてIP電話機に置き換えられる可能性はわずかですが，今後は段階を経ながら，

1）一般電話機同士のIP通話
2）一般電話機とIP電話機同士のIP通話
3）IP電話機同士のIP通話

と，本当の意味でのデータ通話が可能となっていくでしょう．

もちろん携帯性のある電話機についてもこの範疇だと考えることができ，今後はすべての電話機が，いわゆる「SIPデータフォン」としての機能をもつ可能性があります．

〔写真1〕NET&COM2003でのVL Inc.日本スタッフ

また3)の段階になれば「アドレス付与」は必要ですが，050のような番号制度は意味をもたなくなります．

SIP URIのようなメールアドレスライクな形式で充分にことが足りるようになり，すべてはアイコンイメージでオンライン/オフラインを表示するものに置き換えられます．

これにより，いままでの通話のような「一度相手にかけてみないと相手が電話に出られるかどうかわからない」という問題はなくなります．

この段階になると，初めて互いに音声以外のデータをダイレクトにやり取りできるようになり，より一段進んだコミュニケーションを体験することができるようになります．

現在，ソフトフォンで実現しているインターネット電話などはこの前例となります．

おかざき・まさと　VL Inc.（ビーエル・インク）CTO

特集第7章

LinuxによりVoIPを実現する
オープンソースで作るIP電話

森島史仁/実吉智裕

ここまでの章では，VoIPの技術と規格について解説を行った．そこで本章は，これまでの解説を確認する意味から，OSとしてLinuxを，プロトコルスタックとしてオープンソースのOpenH323を使用し，ARM搭載マイコンボードArmadillo上でVoIPを実現し，実際の動作を確認することにより，VoIPに対する理解を深める．

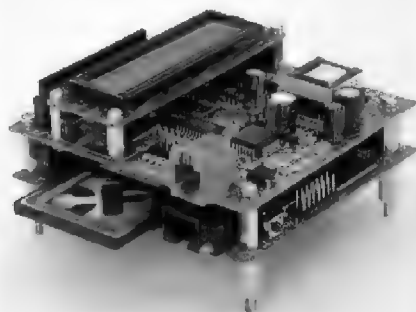
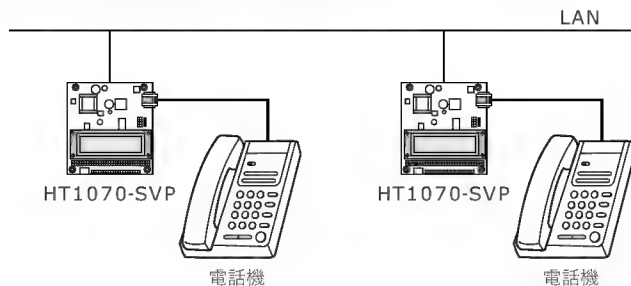
（編集部）

1 はじめに

本稿では，オープンソースを利用して作成された実験用IP電話「VoIP実験キット HT1070-SVP」(http://armadillo.atmark-techno.com/voip/，**写真1**)の開発事例を通して，電話制御ボードの実装とオープンソースのプロトコルスタックOpenH323(http://www.openh323.org/)の利用に関して説明します．

1.1 VoIP実験キットHT1070-SVPの概要

HT1070-SVPは，市販の電話機を接続して使用するタイプの"IP電話実験セット"です(**図1**)．

〔写真1〕VoIP実験キット HT1070-SVP

Linux対応ARM7 CPUボードArmadillo(HT1070-SDK)と電話制御ボード(HT1071-U00)の組み合わせで構成されています．ArmadilloのOSはLinux(kernel 2.4.16)を使用しています．VoIPの通信プロトコルとしてITU-T H.323，そのプロトコルスタックとしてOpenH323を使用しています．音声コーデックはG.711 μ-Law(64kbps)です．

ジッタバッファや無音声圧縮などの各種設定を変更し，実験できるのが特徴です．実験のプログラムは付属のCD-ROMで供給されており，Armadilloへ挿し込んだコンパクトフラッシュにプログラムを格納して使用します(**図2**)．

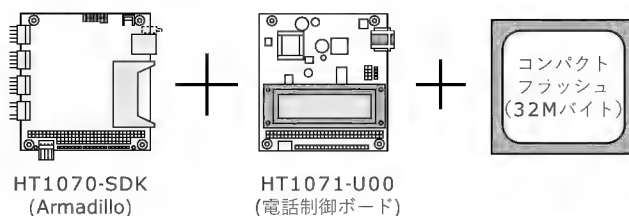
- **Armadillo(HT1070-SDK)**

プロセッサにCS89712(Cirrus Logic製，ARM720Tコア)を採用したPC/104規格サイズのCPUボードです．OSとしてLinuxを搭載しており，標準でLAN(10Base-T)も搭載しています．コンパクトフラッシュをハードディスクと同じように使うことができるため，大きなデータも扱えます．HT1070-SVPではソフトウェア(OS，アプリケーション)をコンパクトフラッシュに入れて動作させています．Armadilloについての詳細は，本誌2002年7月号を参照してください．

- **電話制御ボード(HT1071-U00)**

市販の電話機を接続・制御をするためのボードです．Armadilloと同じくPC/104規格サイズです．Armadilloから5Vと12Vの電源を供給し，電話機の制御に必要な機能をもっています(**写真2**)．

〔図1〕IP電話の実験をするための接続

〔図2〕VoIP実験セットの構成内容

2 ハードウェア

2.1 ブロック図

図3にHT1070-SVPのブロック図を示します．

Armadillo基板と電話制御基板の間はPC/104バスで接続されているほか，音声データを通信するためのディジタルオーディオインターフェースの信号が接続されています．HT1070-SVPには5Vと12Vの電源が必要です．

2.2 ハードウェア各機能

● SLIC（インターシル：HC5518x）

SLICは加入者線インターフェース回路です．実際の加入者線と同じように，直接電話機やFAXを接続することができます．加入者線への給電，ハイブリッド（2線4線変換），リンギング（鳴動），極性反転，監視などの機能をもちます．

SLICは2種類の電源（－20V，－80V）を使用し，制御線（SLICのE0，F0～3信号）の設定により動作モードを切り替えることができます．動作モードには順方向アクティブ，反転方向アクティブ，リンギング，Tipオープン，スタンバイ（ループ検出はアクティブのまま），パワーダウンがあります．

リンギング動作には16Hzのクロックを必要とし，これはCPLDから供給されます．加入者線のループ電流によりオンフック/オフフックを検出し，割り込みを発生します．

● DTMFレシーバ（東芝：TC35302）

東芝のTC35302を使用し，プッシュ回線に使われているボタントーンの音声をデコードします．電話機からのトーン入力を検出すると割り込みを発生します．

● CODEC（モトローラ：MC14LC5480）

汎用の単機能CODECを使用し，音声のアナログ-ディジタル変換を行っています．符号化方式はPCM64kbps（G.711，μ-Law）です．

● CPLD（ザイリンクス：XC9572XL）

CPLD（Complex Programable Logic Device）では，次の各種機能を実装しています．

- PC/104バスの制御インターフェース
- Armadilloのディジタルオーディオインターフェースと電話制御ボードのPCMデータインターフェースの信号変換
- 8kHzから16Hzを生成（リング用）
- 割り込みの制御（フック，DTMF検出）
- DTMFレシーバ，液晶モジュールへの制御線を生成
- SLICの制御線を生成

● 液晶（SC1602）

16桁×2行の表示ができ，各種ステータスを表示します．

● 電源ブロック

電話制御ボードの電源は，PC/104バスから5V，12Vを供給されますが，あらかじめArmadilloに5Vと12Vを供給しておく必要があります．また5Vから3.3Vを生成します．さらに12Vからー20Vおよびー80Vを生成しSLICに供給します（Armadillo単体では5Vで動作する）．

〔写真2〕Armadillo（左）と電話制御ボード（右）

2.3 音声の流れ（ディジタル部分）

● DAI-CPLD間のフォーマット

Armadilloで使用しているプロセッサCS89712はディジタルオーディオインターフェースを搭載しています．本来はオーディオCDと同じサンプリング周波数44.1kHz，データ幅16ビット，2チャネルというフォーマットの入出力ができるものです．HT1070-SVPではサンプリング周波数8kHzで使用し，データ幅16ビット中の上位8ビット，1チャネルのみを使用しています（**図4**，p.119）．DAIのデータは先詰め16ビット，64fs（サンプリング周波数の64倍）のフォーマットで入出力し，電話制御ボードのCPLDと接続します．

● CPLD-CODEC間のフォーマット

HT1070-SVPのCODEC（MC14LC5480）はPCMデータインターフェースのLong Frameフォーマットを使用しています．PCMCLK（ビットクロック）を基準に8ビット幅のFSYNC（同期フレーム）がアクティブになっているときにデータを入出力します．

● CPLDの役割

CPLDでは電圧のレベル変換と，ディジタルオーディオインターフェースとPCMデータインターフェースの信号変換を行っています．

2.4 音声の流れ（アナログ部分）

● CODEC-SLIC-電話機

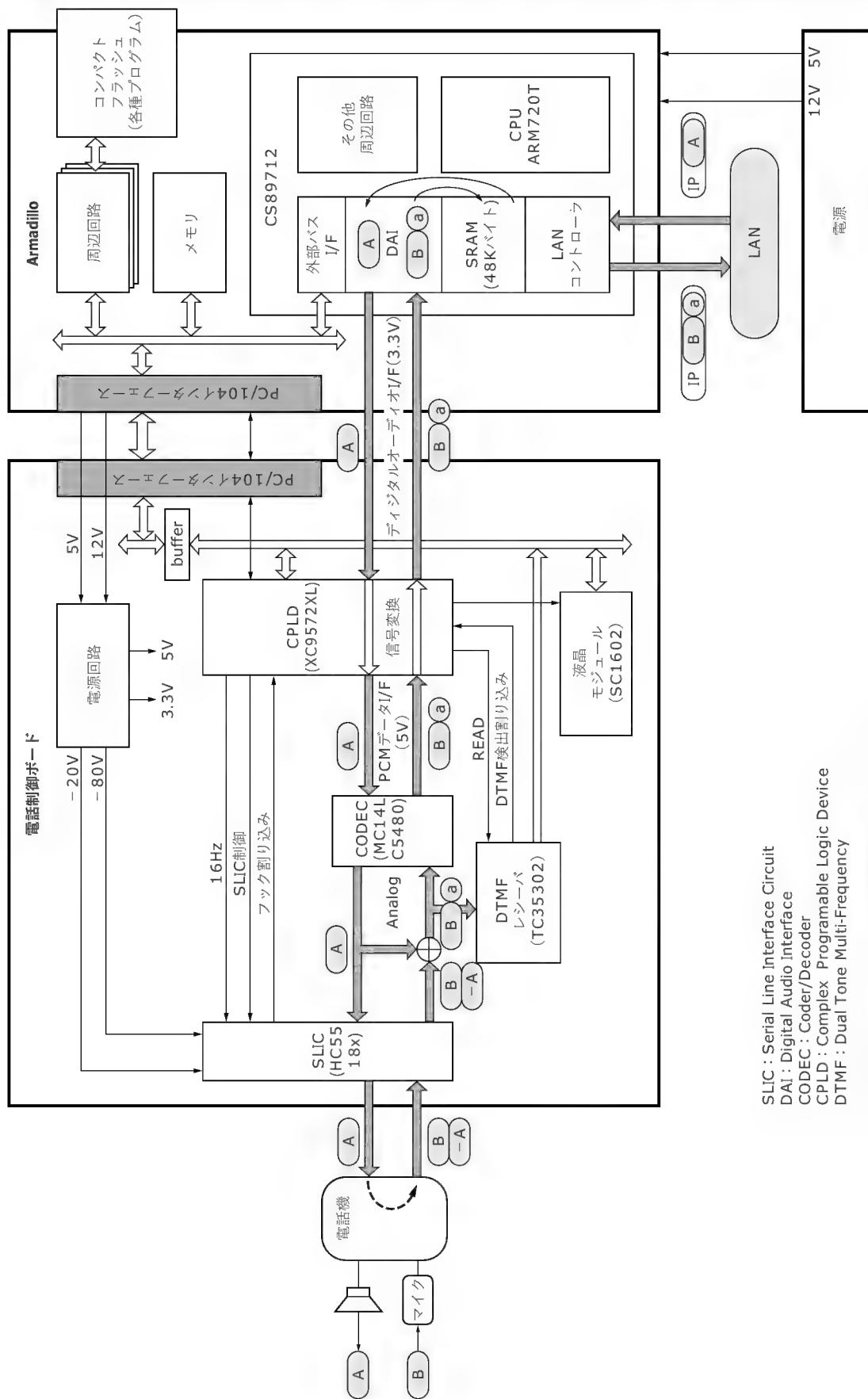
CPUから電話機への音声をAとします．電話機からCPUへの音声をBとします．CODEC（MC14LC5480）ではPCM（G.711，μ-Law，64kbps）とアナログの信号変換が行われます．

CODECから出力されるAは，SLICにより電話機の信号レベルに変換されます．SLICから電話機に出力されたAには，電話機のマイクから入力されたBが合成され，再びSLICを通しCODECに向かって入力されます．このとき，Aの信号は反転しーAになります．SLICからCODECまでの間で，ーAとAは合成され，CODECに入力されるときには，Bのみとなります．

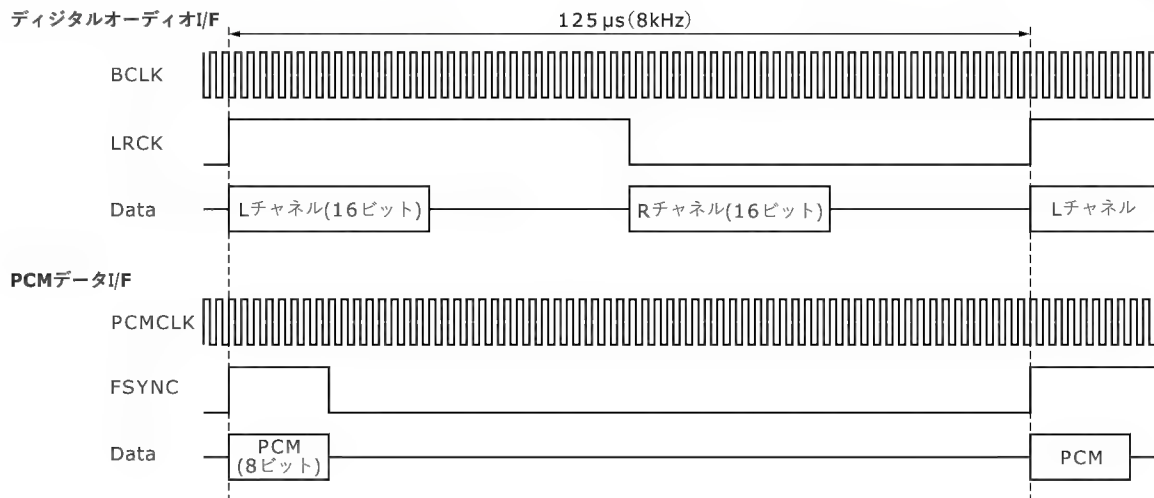
● エコーの問題

しかしながら，実際にはーAとAを合成しても完全に消去する

〔図3〕HT1070-SVPのハードウェアブロック図

電話制御ボード
−20V
−80V
電源回路
5V
12V
3.3V　5V
buffer
PC/104インターフェース
16Hz
SLIC制御
フック割り込み
CPLD
(XC9572XL)
信号変換
SLIC
(HC55
18x)
CODEC
(MC14L
C5480)
Analog
PCMデータI/F
(5V)
READ
DTMF検出割り込み
DTMF
レシーバ
(TC35302)
液晶
モジュール
(SC1602)
電話機
マイク
A
B
a
−A
ディジタルオーディオI/F(3.3V)
Armadillo
PC/104インターフェース
周辺回路
コンパクト
フラッシュ
(各種プログラム)
メモリ
CS89712
外部バス
I/F
DAI
SRAM
(48Kバイト)
LAN
コントローラ
その他
周辺回路
CPU
ARM720T
IP
LAN
12V　5V
電源

SLIC : Serial Line Interface Circuit
DAI : Digital Audio Interface
CODEC : Coder/Decoder
CPLD : Complex Programable Logic Device
DTMF : Dual Tone Multi-Frequency

※電話機－SLIC間の通信は実際の電気的な信号とは異なるが，
音声の流れがわかりやすくなるような図にしている

〔図4〕ディジタルオーディオI/FとPCMデータI/Fの対応

ことはできず音声aが残り，正確にはBとaが相手に返ります．aが残ったままの状態でIP電話として運用すると，自分の話した声がネットワークを経由して遅延して自分に返ってきます．これがエコーです．普通の電話機を使ったIP電話を実現するうえで，このエコーをなくすことが一つの課題としてあげられています．

2.5 Armadillo上での音声の処理

CS89712のDAIは送信で8ワード，受信で12ワードのFIFOをもっており，そこのFIFOの割り込みを使用してデータの送受信を行います．ARMアーキテクチャの割り込みには，FIQ(高速割り込みリクエスト)とIRQ(割り込みリクエスト)の2種類あり，CS89712のDAIはFIQを使用しています．

かりに送信側のFIFOでHalf Emptyの割り込みが起こった場合，125μs(サンプリング周波数8kHz)×4word＝500μsで割り込み応答をする必要があります．

Linuxはリアルタイム OSではないので，通常の割り込み(IRQ)と同じように処理をしていては，この割り込み応答時間を保証することができず，最悪の場合，音が途切れることになります．

しかしながら，Armadillo LinuxにおけるFIQの扱いは，Linuxの通常の割り込み(IRQ)よりも高いレベルで実行され，カーネルすらも止めてまで割り込み応答を行います．そのためArmadillo Linuxではリアルタイム OSではないのにも関わらず，音の途切れることのないデータの入出力が可能です．

HT1070-SVPでは，このFIQでの割り込み処理時間を極力短くするために，音声データのバッファリングをCS89712内蔵のSRAM(48kバイト)で行っています．

3 ソフトウェア

3.1 OpenH323概要

はじめに，オープンソースのプロトコルスタックOpenH323について説明します．

OpenH323はMPL準拠のオープンソースなプロトコルスタックで，WindowsやUNIXなど，各種OS上で動作します．ソースコードはC++で記述されています．

現在OpenH323では，H.323にて定義されている基本的なプロトコルはもちろん，その他多くの機能が実装されています．

基本的な機能

- H.225　：呼制御(Q.931)/ゲートキーパーメッセージ(RAS)
- H.245　：マルチメディア通信の制御
- RTP/RTCP：メディア転送
- H.235　：セキュリティ/暗号化

その他の機能

- H.245トンネリング
- H.235によるゲートキーパーへのアクセス
- G.723.1，G729など，各種音声コーデックへの対応
- H.450シリーズによる付帯サービス
- ジッタバッファ
- 無音圧縮
- 動的な閾値による無音検出
- ブロードキャストとマルチキャストによるゲートキーパー検出機能

また，OpenH323プロジェクトが提供しているアプリケーションには，以下のようなものがあります．

- OhPhone　：コマンドラインベースのIP電話
- OpenPhone：GUIベースのIP電話
- OpenMCU　：電話会議サーバ
- OpenAM　：留守番電話機
- OpenGK　：ゲートキーパー
- PSTNGw　：公衆電話網へのゲートウェイ
- T38Modem：Fax

3.2 OpenH323のインストール

OpenH323をArmadillo用にコンパイルしインストールする

方法を説明します．

Armadillo用にコンパイルするには，あらかじめクロスコンパイル環境を用意する必要があります．クロスコンパイル環境の構築に関しては本誌2002年7月号の記事，または`http://armadillo.atmark-techno.com/`を参照してください．

まずはじめに，OpenH323プロジェクトのWebページからpwlibとopenh323のソースコードをダウンロードします（pwlibはPortable Windows Libraryの略で，openh323がその基底クラスとして利用しているクラスライブラリ）．

そして，クロスコンパイルを行うホストマシンの任意のディレクトリで，以下のようにコマンドを入力してソースコードを展開します．今回は`/usr/local/src`以下に展開するものとして話を進めます．

```
$ tar zxvf  pwlib_1.4.10.tar.gz -C
                              /usr/local/src
$ tar zxvf openh323_1.11.6.tar.gz -C
                              /usr/local/src
```

pwlibとopenh323をコンパイルするための環境変数を設定します．

設定する環境変数は，`PWLIBDIR`と`OPENH323DIR`，`LD_LIBRARY_PATH`で，それぞれ以下のようにコマンドを入力します（bashの場合）．

```
$ export PWLIBDIR=/usr/local/src/pwlib
$ export OPENH323DIR=
                    /usr/local/src/openh323
$ export LD_LIBRARY_PATH=
             $PWLIBDIR/lib:$OPENH323DIR/lib
```

ホストマシン用のasnparser（ASN1の定義からソースコードを作成するプログラム）を作成するために，pwlibのコンパイルを行います．

以下のようにコマンドを入力します．

```
$ cd $PWLIBDIR; make opt
```

クロスコンパイル用に環境変数を設定します．次のようにコマンドを入力します（bashの場合）．

```
$ export MACHTYPE=arm
$ export HOST_PLATFORM_TYPE=linux_x86
```
（ホストマシンがx86の場合）
```
$ export CC=arm-linux-gcc
$ export CPLUS=arm-linux-g++
```

ARMプロセッサ向けにpwlibとopenh323をコンパイルします．

```
$ cd $PWLIBDIR
$ make opt SYSINCDIR=/usr/arm-linux/include
$ cd $OPENH323DIR
$ make opt SYSINCDIR=/usr/arm-linux/include
```

（※`SYSINCDIR`にはクロス開発環境のincludeディレクトリを指定する）

コンパイルが終了したら，`/usr/loca/src/pwlib/lib`の下に作成されたpwlibライブラリ`libpt_linux_arm_r.so.X.X.X`（Xはバージョン番号）と`/usr/local/src/openh323/lib`の下に作成されたOpenH323ライブラリ`libh323_linux_arm_r.so.X.X.X`，そして，`/usr/local/openh323/samples/simple/obj_linux_arm_r`の下に作成されたサンプル用の通話プログラムである`simpH323`をArmadilloにftpでファイル転送し，pwlibライブラリとopenh323ライブラリを`/usr/lib`の下に移動します．

転送したライブラリファイルのシンボリックリンクを作成します．

Armadillo上で`/usr/lib`に移動し，次のようにコマンドを入力してください．

```
$ ln -s libpt_linux_arm_r.so.X.X.X
                  libpt_linux_arm_r.so.1
$ ln -s libh323_linux_arm_r.so.X.X.X
                libh323_linux_arm_n.so.1
```

最後に`SimpH`に実行権を与えて，実行します．無事に起動されることを確認します．ただし，オーディオデバイスが未実装の場合，通話はできません．

3.3　OpenH323を利用したIP電話の作成方法

OpenH323を利用してオリジナルのIP電話アプリケーションを作成する方法を，HT1070-SVPを例にして説明します．

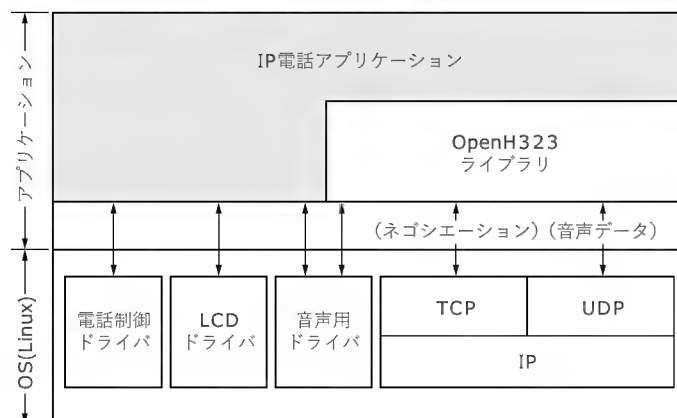
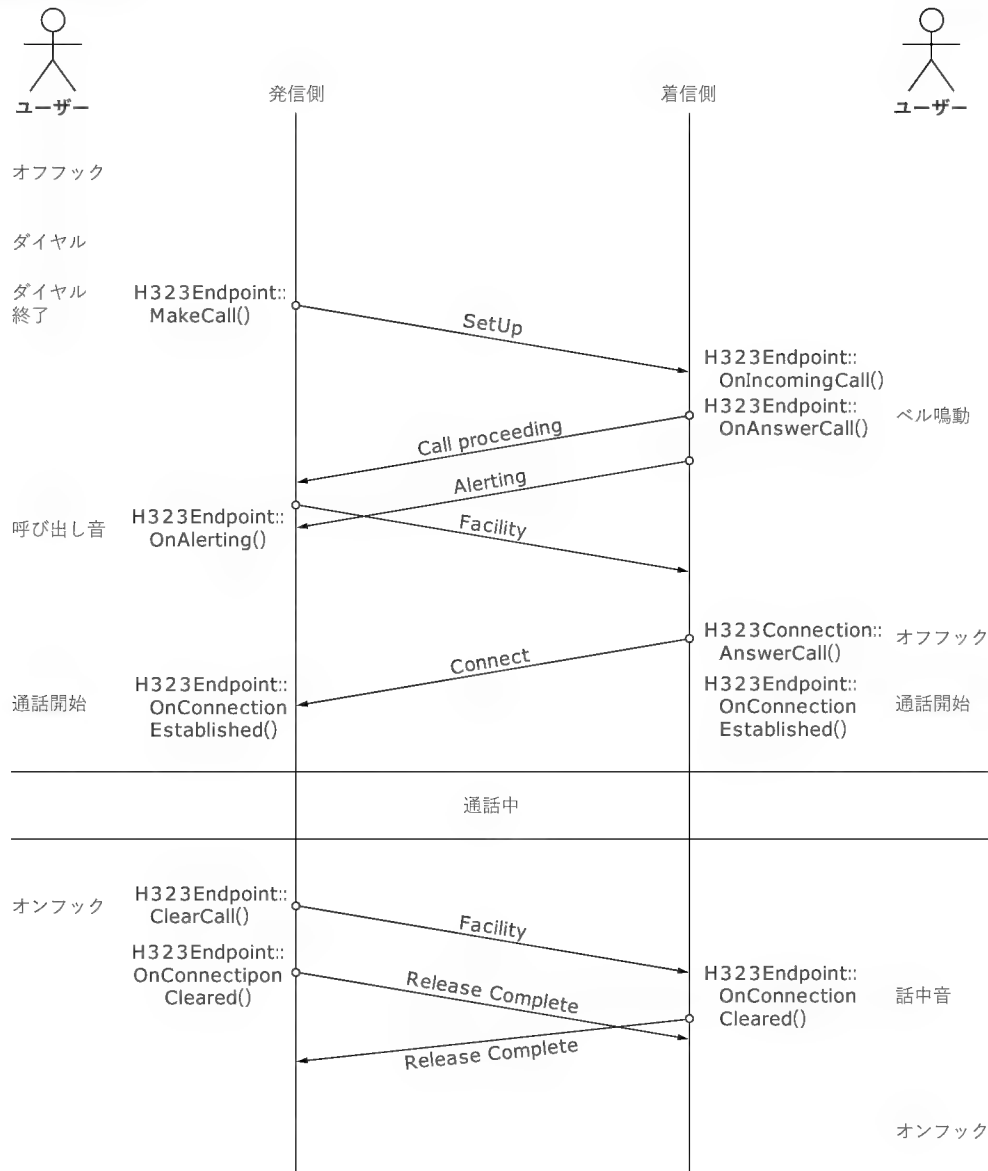
はじめにHT1070-SVPのソフトウェア構成図を示します（**図5**）．

HT1070-SVPでは，IP電話アプリケーションが，電話の状態管理とLCDの制御を行い，OpenH323ライブラリがH.323によるIP通信全般を行っています．

OpenH323ライブラリを利用してオリジナルIP電話を作成するには，少なくともH323Endpointクラスの派生クラスを作成し，各種コールバック関数をオーバライドする必要があります．

発信から切断までの動作フローと，その間に呼び出される主要な関数の関係は，**図6**のようになります．

図6で紹介した主要な関数の処理概要とHT1070-SVPにおける実装を**表1**（p.122）に説明します．

〔図5〕HT1070-SVPのソフトウェア構成図

〔図6〕発信から切断までの動作フロー

3.4 電話機の制御

最後に電話機の制御に関するいくつかのポイントを説明します.

● **フックの検出**

フック状態が変化すると，SLICデバイスにより検出されて，イベントが通知されます(**図7**).

オフフック時は問題ありませんが，オンフック時は，公衆網の定義において0.3秒以内のオンフック状態は瞬断とみなされるので(正確には0.1秒以内は瞬断，0.1～0.3秒は不定)，オンフック発生時は，その後の0.3秒間にオフフックが発生しなかった場合のみ，オンフックとみなして処理しています.

● **可聴音の作成**

リスト1(p.123)の式を使用して発信音，話中音(400Hz)と呼出音(400Hzの16Hz変調)を作成しています.

発信音は400Hzをそのまま流し，話中音は400HzをタイマでON/OFFしながら流しています．また，呼び出し音も400Hzの16Hz変調音をタイマでON/OFFしながら流す実装にしています.

● **呼び出し信号(リング)(図8)**

呼び出し信号はSLICデバイスにて発生させます．ソフトウェアではタイマを使用し，一定周期でSLICデバイスに対し呼出信号の発生と停止を行うことで，公衆網の電話と同様の周期でベルを鳴動させています.

● **ダイヤル動作の完了(図9，p.123)**

ユーザーが相手先番号をダイヤルするとき，ダイヤルの完了をどのように判断するかが問題となります．HT1070-SVPでは，オフフック後，最初のダイヤルが押されると適切な時間のタイマを設定し，その後のダイヤルが押されるたびに前のタイマを解除して再度タイマを設定します．そしてタイムアウトが発生した場合に，ユーザーのダイヤルが終了したとみなして，発信

〔表1〕主要な関数の処理概要とHT1070-SVPにおける実装

▶ H323Endpoint::MakeCall()

処理概要	引き数で指定されたアドレスに対して発信を行う
H1070-SVPでの実装	ユーザーのダイヤル動作終了時，ダイヤルで指定されたアドレスを指定して関数を呼出し，発信を行う

▶ H323Endpoint::OnIncomingCall()

処理概要	Setupメッセージ受信時のコールバック関数．Alertingメッセージを送信する前に呼出される．戻り値がTRUEの場合はネゴシエーションを継続，FALSEの場合ネゴシエーションを中断してRelease Completeメッセージを送信する
H1070-SVPでの実装	関数をオーバライドし，電話機のフック状態がオフフックであれば，相手にビジィを伝えるためにFALSEを返し，オンフックの場合はTRUEを返してネゴシエーションを継続する

▶ H323Endpoint::OnAnswerCall()

処理概要	Setupメッセージ受信時のコールバック関数 Connectメッセージを送信する前に呼出される．戻り値で，“ネゴシエーションの中断”か，“コネクションの確立”，または“ユーザーによるコネクション確立タイミングの指定”のいずれかを指定する
H1070-SVPでの実装	関数をオーバライドし，電話機のベルを鳴らし始め，戻り値は“ユーザーによるコネクション確立タイミングの指定”(AnswerCallPending)を返し，ユーザーのオフフックを待つ

▶ H323Endpoint::OnAlerting()

処理概要	Alertingメッセージ受信時のコールバック関数 戻り値がTRUEの場合はネゴシエーションを継続，FALSEの場合ネゴシエーションを中断してReleaseCompleteメッセージを送信する
H1070-SVPでの実装	関数をオーバライドし，呼出音を発生させて，ユーザーに呼出中であることを伝える

▶ H323Connection::AnswerCall()

処理概要	Connectメッセージを送信する
H1070-SVPでの実装	ベル鳴動中に電話がオフフックされた場合，この関数を呼出して通話を確立させる

▶ H323Endpoint::OnConnectionEstablished()

処理概要	コネクション接続時のコールバック関数．接続されたコネクションに関する情報が引き数で渡される
H1070-SVPでの実装	関数をオーバライドして，発信側だった場合には呼出音を止める

▶ H323Endpoint::ClearCall()

処理概要	コネクションを切断し，通話を終了させる
H1070-SVPでの実装	通話状態のときに電話がオンフックされた場合，この関数を呼出して通話を終了させる

▶ H323Endpoint::OnConnectionCleared()

処理概要	コネクション切断時のコールバック関数．切断されたコネクションに関する情報が引き数で渡される
H1070-SVPでの実装	関数をオーバライドし，被切断側だった場合は，話中音を流してユーザーに通話の終了を知らせる

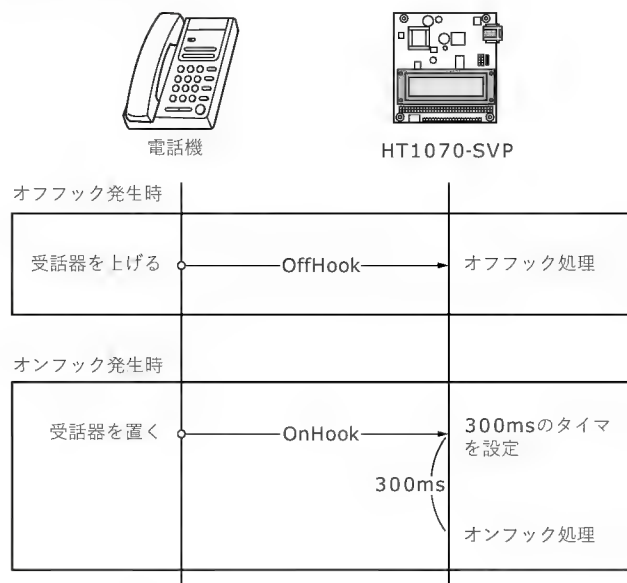
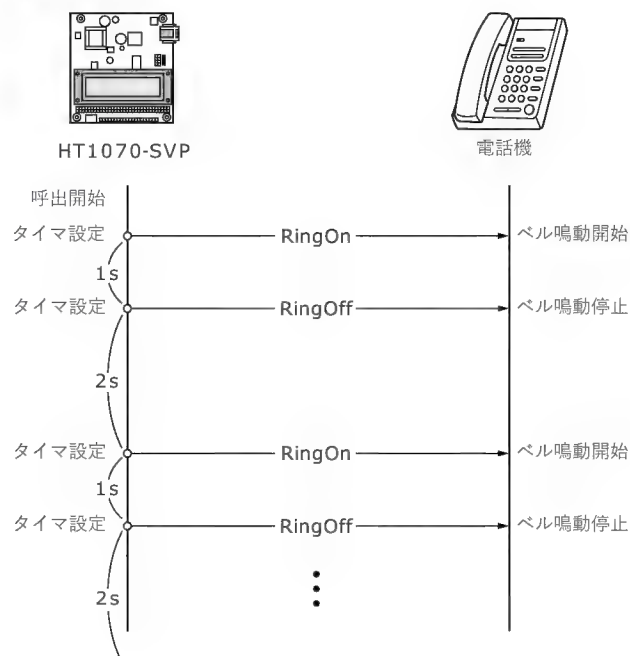
動作に移ります．

● **IPアドレスのダイヤル方法**

IP電話では，通話先の指定は最終的にIPアドレスで行われます．

HT1070-SVPでは，普通の番号とIPアドレスの変換テーブルを用意し，番号で電話をかけられるようにしていますが，直接IPアドレスを指定する方法も用意しています．

直接IPアドレスを指定する場合は，最初に“＊”を押し，その後“.”の代わりに“＃”を押します．たとえば192.168.1.20に電話をかける場合なら，“＊198＃168＃1＃20”とダイヤルします．

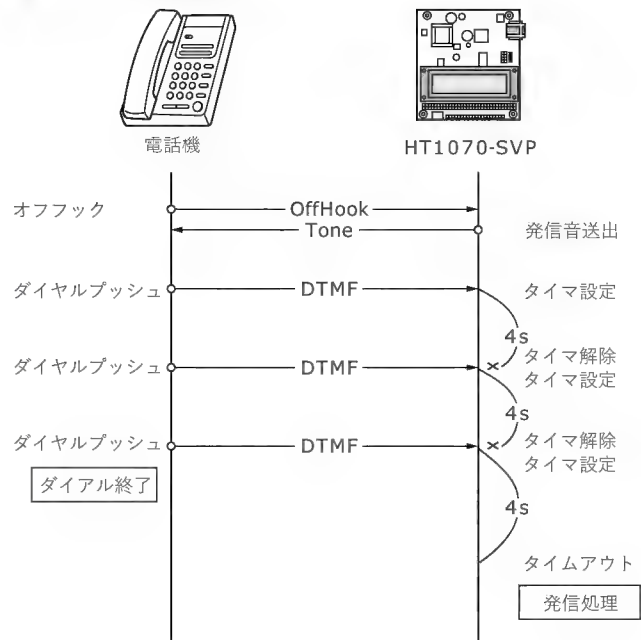
〔図7〕フックの検出

〔図8〕呼出信号

〔リスト1〕可聴音の作成

```
char tone_data[BUF_SIZE];    // 発信，話中音データ
char ring_data[BUF_SIZE];    // 呼出音データ

const int TONE_HZ   = 400;  // 400Hz(発信音周波数)
const int MODU_HZ   = 16;   // 16Hz(変調周波数)
const int SAMP_HZ   = 8000; // 8kHz(サンプリングレート)

const double coneffcient = M_PI * 2 / SAMP_HZ;

for(int i = 0 ; i < BUF_SIZE ; i++){
  tone_data[i] = (short)(SHRT_MAX * sin(TONE_HZ * coefficient * i));
  ring_data[i] = (short)(tone_data[i] * sin(MODU_HZ * coefficient
                                                          * i));
}
```

おわりに

OpenH323のようなオープンソースのプロトコルライブラリのおかげで，プロトコルの実装という敷居が高い開発をせずに，気軽にVoIPアプリケーションを作成できるということがおわかりいただけたかと思います．

ぜひ，世界に一つのオリジナルIP電話を作ってみてください．

〔図9〕ダイヤル動作の完了

参考文献

1）OpenH323プロジェクト，http://www.openh323.org/
2）技術参考資料「電話サービスのインタフェース(第5版)」，日本電信電話(株)
3）特集関連「Armadilloの概要と使用方法」，『Interface』，2002年7月号
4）Armadillo公式サイト，http://armadillo.atmark-techno.com/

もりしま・ふみと/さねよし・ともひろ　(株)アットマークテクノ

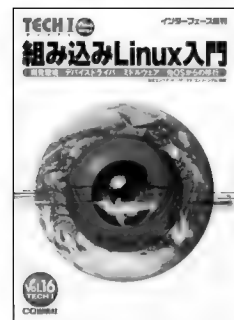
TECH I Vol.16 (Interface4月号増刊)　好評発売中

組み込みLinux入門

開発環境/デバイスドライバ/ミドルウェア/他OSからの移行

日本エンベデッドリナックスコンソーシアム 監修
B5判　272ページ　定価2,200円(税込)

サーバ用途でかなり普及したLinuxだが，組み込みシステム開発へのLinux導入の取り組みも着々と進行している．

本書では，組み込みシステムの開発にLinuxを使うための技術要素を，入門者向けに，総覧的に解説している．内容としては，組み込みLinuxの現状，開発環境，カーネル/デバイスドライバ，ミドルウェア，他OSからの移行などを盛り込んでいる．また，組み込みLinuxに関連するキーマンへのインタビューも収録している．

CQ出版社　〒170-8461 東京都豊島区巣鴨1-14-2　販売部 TEL.03-5395-2141　振替 00100-7-10665

組み込みプログラミングノウハウ入門

第12回 メッセージベーススケジューリングと実装
―A Practitioner's Handbook for Real-Time Analysisを読む

藤倉俊幸

はじめに

今回は，複数のタスクがメッセージパッシングにより連携して処理を行うような場合のタイミング解析を取り上げる．メッセージパッシングは，リアルタイムOS（RTOS）を利用したほとんどの組み込みシステムで使われる実現方式である．

● シナリオによる設計

最近はオブジェクト指向による分析設計も普及期に入り，著者のところにも，以前より実践的な内容での問い合わせが増えてきたように思う．その一つが，ここで取り上げるメッセージベースで設計した場合のタイミング解析である．

オブジェクト指向で分析設計を行った場合の仕事の流れは，次のようになる．

- アクタの抽出
- ユースケースの抽出
- シナリオの抽出

ここで出てくるアクタ・ユースケース・シナリオが基本的なセットになる．そして，分析・設計・実装と進むうちに，これらが姿を変えていく．具体的な例としては，ユースケースがコラボレーション図になり，シナリオがシーケンス図になる．つまり，オブジェクト指向を使った場合，要求仕様や設計仕様としてシーケンス図を使う機会が増える．シーケンス図からはダイレクトにメッセージベースの設計・実装に変換可能である．しかし，単純に変換するだけでは，本当に動くかどうかは誰にもわからない．

アクティブオブジェクトを使った場合のシーケンス図の例を**図1**に示す．たとえばこの図で，callerからexchangeがlift receiverメッセージを受け取ったら，exchangeは1秒以内にdial toneを返さなくてはならない，と記述してある．このほかにもいくつかの時間制約が記述してある．このような時間制約を守ることができるのか，できるとすればどのようなアーキテクチャを選べばよいのか，どのような設計をすればよいのか，どのような実装をすればよいのかが問題となる．アーキテクチャか設計か実装か，どのレベルで対応できるのかを知ることも重要である．

図1が，分析の後半か設計の前半，すなわちアーキテクチャレベルであれば，タイミング解析が必要になる．設計の後半か実装段階であればlift receiverによって呼び出される関数の実行時間を1秒以内におさえればよいことになる．

〔**図1**〕**シーケンス図の例**

（*OMG-Unified Modeling Language*, pp. 3-104, v1.4 September 2001より）

前者と後者の決定的な違いは，前者はマルチタスク環境下での仕様，後者はCPUを占有して実行した場合の仕様だということである．設計の過程で，前者から後者に仕様を具体化することが必須になってくる．ここをおさえていないと，まともに動くかどうかは現場に設置してオーバロードになるまでわからないことになる．

また，たまたまオーバロードが発生して時間制約が守れないエラーが検出されても，数十万回に1回起こるハードウェアの誤動作なのか区別が付かなくなってしまう．エラーが起こるとすればどのようなとき，どの程度の確率で起こるか，コントロール下にシステムを置く必要がある．本当に動くかどうかわかる状態にしておくということである．

● シーケンス図

ユースケースを使わない場合でも，あるいはオ

ブジェクト指向以前であっても，シーケンス図ライクなものはソフトウェアを構成するものの間の動的な関係を記述するのに便利なので，利用価値が高い．

シーケンス図を描いたとき，上に並ぶものを「インタラクティブインスタンス」と呼ぶが，これがオブジェクトであるか，モジュールであるか，サブシステムであるか，システムとアクタであるか，タスクであるかによって抽象度が変わるが，表現される動き自身は，一般にはかなり具体的なものである．1枚のシーケンス図は，そのインタラクティブインスタンスの抽象度における動作例とかエピソードに対応する．1枚で一つの動作例とかエピソードなので，シーケンス図1枚だけではあまり意味がなくて，仕様的には正常系・異常系・例外処理などを含めた複数枚でセットになる．

組み込みプログラミングでは並行処理が前提になるので，これらの複数のシーケンス図が同時にアクティブになることがあるのか，なったとしたらどうなるのかを考えなければならない．同時に起動される場合に考えるべきことは，排他制御の方法やデッドロックの有無，トリガとなる外部イベントにデッドラインが設定されていた場合にそのデッドラインを守れるかどうかなどである．

オブジェクト指向をまったく使わない場合は，上に並ぶものはインタラクティブインスタンスではなく個々のアクションと見ることもできる．この場合は，外部からのイベント入力がどのようなアクション列で処理されるか示すことになる．このような解釈がもっとも抽象的かもしれない．この解釈に立てば，自由度が広がる．

● A Practitioner's Handbook for Real-Time Analysis

自由度を広げたところで，手元の *A Practitioner's Handbook for Real-Time Analysis* [1]（**図2**）を見てみたらちょうど良い例が載っていたので，これを利用することにする．アーキテクチャレベルの仕様を確認し詳細設計レベルの仕様に具体化する例としてちょうど良いという意味である．

図3から**図8**は，その例（pp.6-66）をシーケンス図で表したものである．各イベントの周期とデッドラインはマルチタスク環境下の話で，各アクションの実行時間はCPUを占有して実行した場合の実行時間である．各イベントのデッドラインを守れることがわかれば，各アクションの実行時間が実装目標になる．システム全体としてみたときのイベントに対する応答時間を，個々の関数の実行時間に具体化している．仕様を具体化するとは，このことである．関数レベルの実行時間であれば，個々の開発者がICEなどを使って計測できる範囲である．

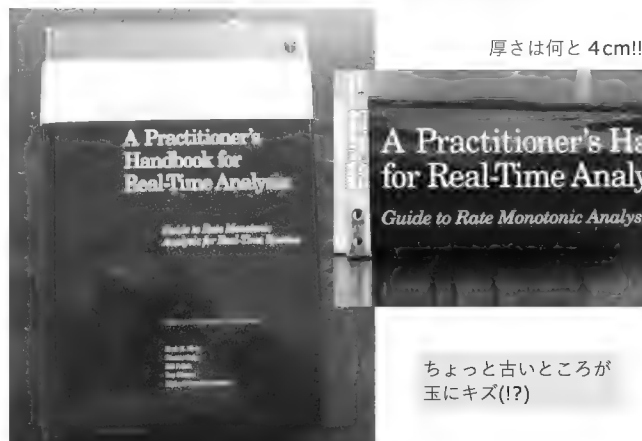
このハンドブック[1]は，Rate Monotonic法をベースとしていろいろな状況でどのようにタイミング解析をするか解説してある貴重な本である．前回取り上げたスポラディックサーバも載っている．組み込みエンジニアであれば手元に置いておくべき本ではないかと思う．ただし，700ページ以上ある，厚さも4cmに迫るので，なかなか簡単に読めるものでもない．読めるとすればまじめな学生さんぐらいではないだろうか．だが，値段も半端ではないので，学生さんにはこの点が辛いかもしれない．この記事によって，どのような例が載っているのかわかれば，お金と時間を投資すべきか判断しやすいと思う．

〔図2〕A Practitioner's Handbook for Real-Time Analysis

図3から**図8**の各イベントは，それぞれの周期で独立に外部から入力される．あとで示す**図10**と**図18**では，アクタを一つしか描いていないが，独立に入力されるのでイベントの数だけアクタを並べたほうがよいかもしれない．あるいは，アクタに多重度をつける手もある．

これらのイベントは，周期と等しいハードデッドラインをもっている．イベントを処理するためのアクションがシーケンス図の上に並んでいる．これを何にするのかが，設計のテーマである．時間の単位は任意だが，コンテキストスイッチングなどのRTOSによるオーバヘッドを無視できるms単位と考えておく．一般にコンテキストスイッチングは，100μs程度と見積もることができる．

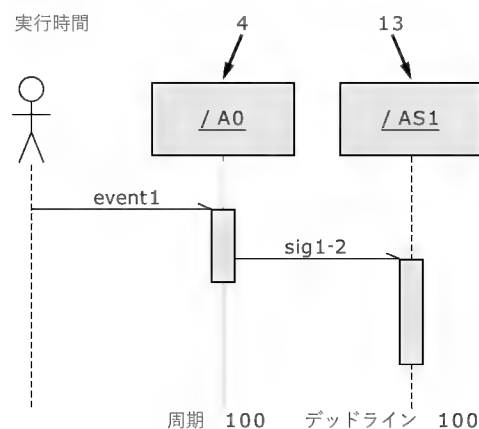
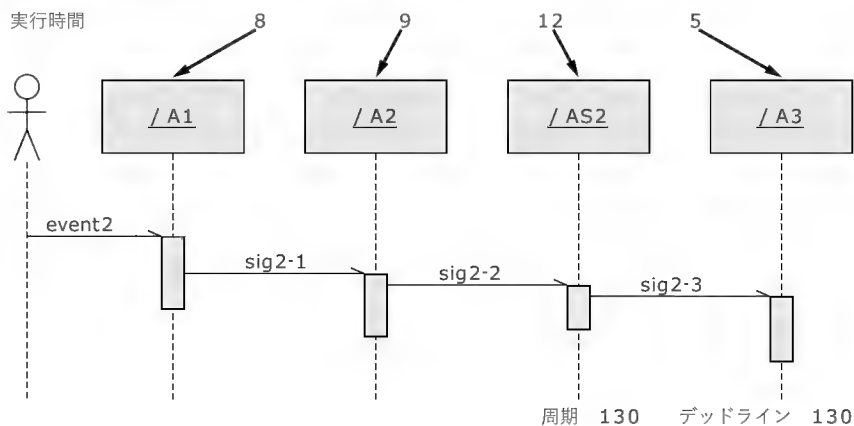
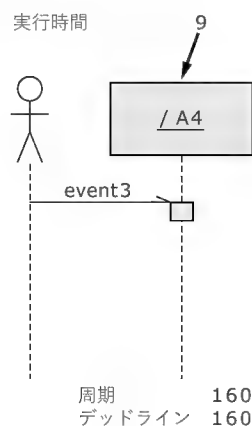
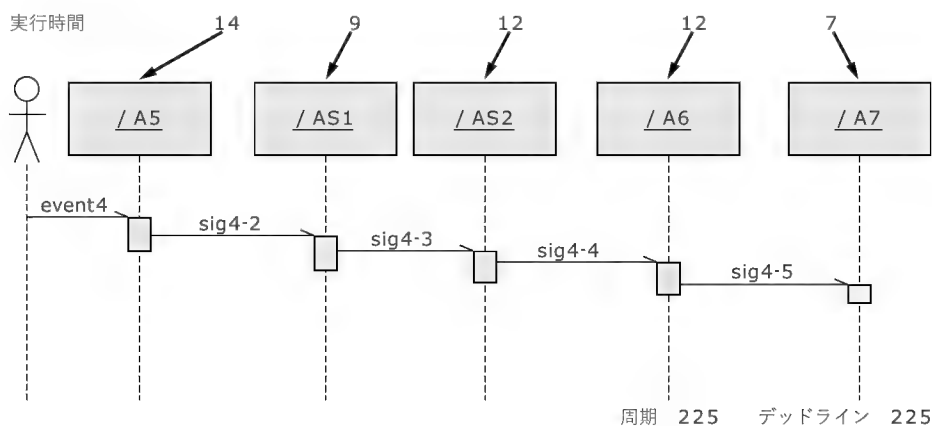
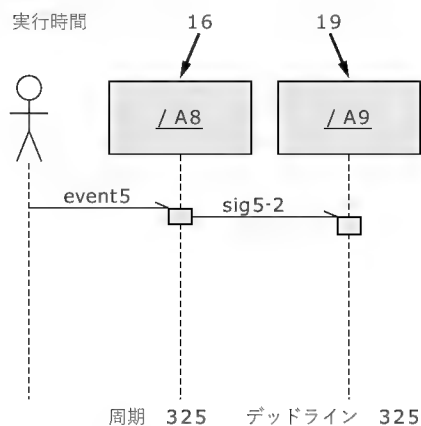
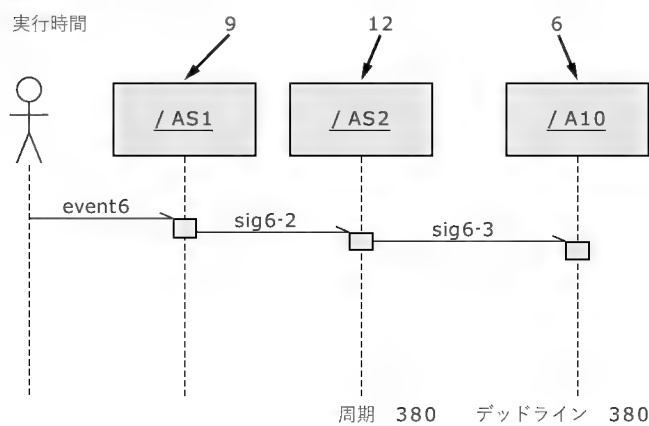
デッドラインが入力周期よりも長い場合は，前のイベントを処理中に次のイベントが入力される場合を考慮しなければならない．その場合，ここでの扱いとは別の，入力をキューイングする方法が必要になる．このようなデッドラインが長いケースについては別途扱いたい（もちろんハンドブックにはそのような例も載っている）．

1 実装方法

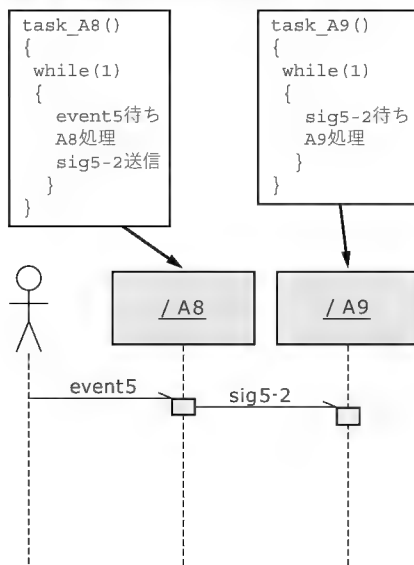
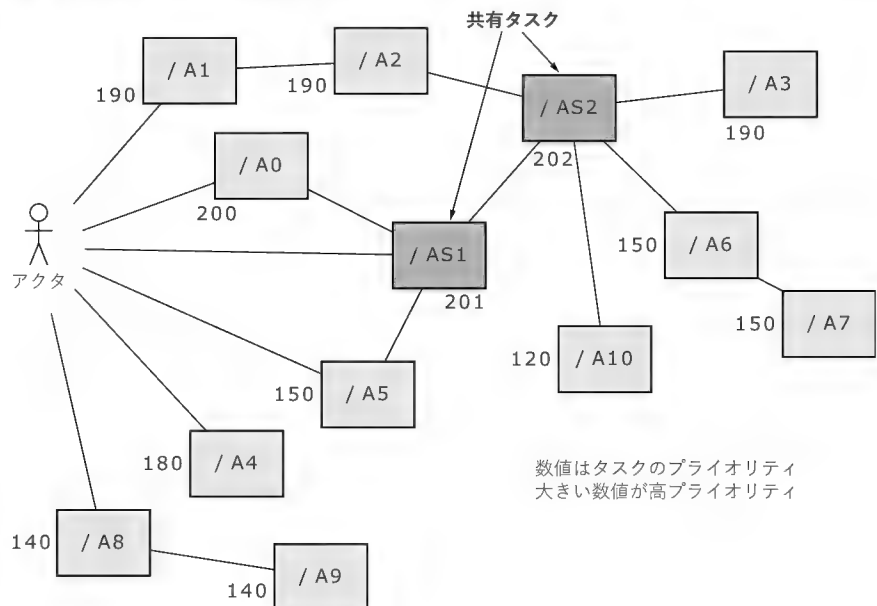
外部仕様としての6種類のイベントに対して，それを処理するために抽出された設計仕様としての13種類のアクションをどのように実装するかによって，タイミング解析の方法も変わる．そして，応答時間も大きく変化する．

● 個別タスクによる実装

最初の実装方法は，アクションすべてを単独のタスクとして実装する方法である．実装のスケルトンを**図9**（p.127）に示す．**図9**はイベント5に対するものだが，他のタスクでもスケルトン

〔図3〕イベント1

〔図4〕イベント2

〔図5〕イベント3

〔図6〕イベント4

〔図7〕イベント5

〔図8〕イベント6

は同一である．ただし，複数のイベントで共有されるアクションAS1とAS2に対応するタスクについては複数メッセージ待ちと，起動メッセージに応じた次のメッセージを送信する実装が必要になるので，もう少し複雑になる(**図10**)．

各タスクのプライオリティは，起動される元イベントの入力周期にしたがってレートモノトニックに決定される．すなわち，短い周期のイベント1に関連したタスクのプライオリティがもっとも高い．共有タスクについては，共有されるイベント系列の中でもっとも高いプライオリティより1だけ高いプライオリティを割り当てる．すなわち，task_AS1ではtask_A0とtask_A5とイベント6により起動されるが，この中でもっともプライオリティが高いのはtask_A0の200なのでtask_AS1のプライオリティは201とする．同様にtask_AS2の場合は，task_AS1からメッセージを受けているので202とする．

〔図9〕個別タスクによる実装

〔図10〕個別タスクによる実装(その2)

このようにプライオリティを決めることで，共有タスクの同期処理をRTOSにまかせることができる．高優先度ロック(highest locker protocol)として機能するのである．

この実装の特徴は，アクション自身の仕様と**図3**から**図8**の各イベントの仕様を一緒に実装している点にある．一般には，あまり良い実装ではない．出所が別の仕様は別々に実装するほうがよい．一般には別々に実装すると部品性は良くなるが，インターフェースが必要になる分だけ余計なオーバヘッドが増えるので，応答性も悪くなると信じられている．しかし，マルチタスク環境で並行実行させる場合には，必ずしもこの原則どおりにならない例が多く存在する．この実装もその例である．この実装方法ではデッドラインを守れないが，次に紹介するサーバタスク方式では，タスク数は増えるが応答性は改善される．

起動周期と処理時間とプライオリティが決まればタイミング解析が可能になる．この場合は，Fixed priority scheduling of periodic tasks with varying execution priority [2)]の手法を使う．この方法も，前述したハンドブックに載っている．ハンドブックの解析手順のほうが，オリジナルの論文よりも少し整理されているように思う．**図11**に計算の概要を示す(普通，状態図をこのようには使わない)．

▶ Step1　標準形への変換

一つの外部イベントに対して複数のタスクが起動されるが，それらのタスクのプライオリティを降順にならないように変換する．たとえば，イベント4の場合，task_A5，task_AS1，task_AS2，task_A6，task_A7の順に起動され，プライオリティは150，201，202，150，150の順で変化する．これを，すべて150にする．アルゴリズムを書くと，次のようになる．

```
j = m(i)
loop until (j ≦ 1)
  if(Pi, j-1 > Pi, j) then Pi, j-1 = Pi, j
  j = j - 1
end loop
```

ここで，$m(i)$はi番目のイベントのアクション数である．たとえば，$m(4)=5$である．$P_{i,j}$は，i番目のイベントのj番目のアクションのプライオリティである．プライオリティを変換後，同一のプライオリティをもった連続したアクションを一つのアクションと考える．この変換をしたタスク列を標準形(canonical form)と呼ぶ(**図12**)．標準形に変換しても，イベント全体の応答時間は変化しないことが証明されている．その意味としては，途中にプライオリティの低いタスクがあればそこがボトルネックになるので，その前までプライオリティが高くても関係ないということである．変換の目的は，標準形にすることで計算を単純にすることである．

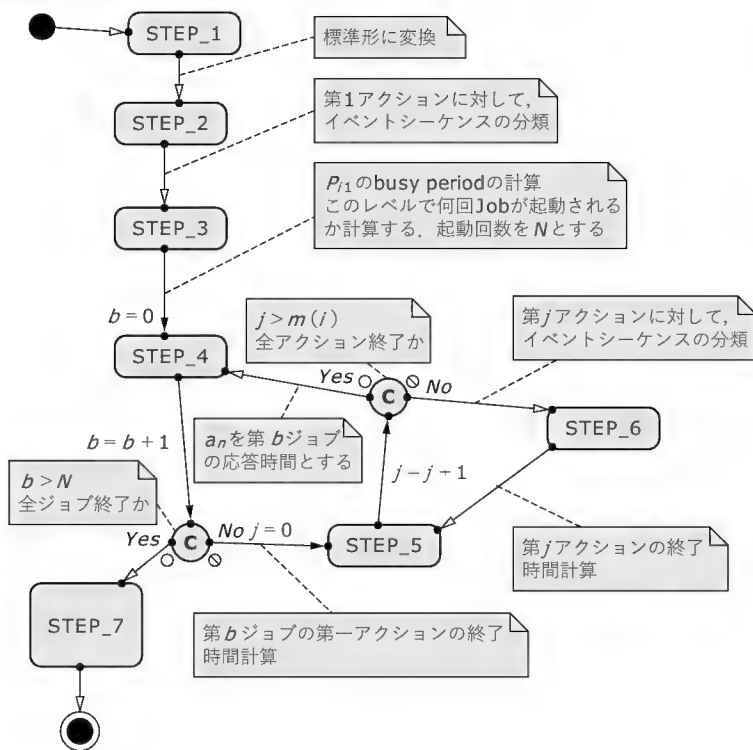
▶ Step2　第1アクションに対するイベントの分類

次に，応答時間を求めようとするイベント以外のイベントを分類する．求めようとしているイベントをi番目のイベントとする．このイベントの第1アクションのプライオリティを$P_{i,1}$とする．この$P_{i,1}$に対して，$i \neq k$であるk番目のイベントについて，

- $H(1)$　：すべてのプライオリティ($P_{k,1}$，$P_{k,2}$，$P_{k,3}$，…)が$P_{i,1}$以上である
- $HL(1)$　：$P_{k,1}$は$P_{i,1}$以上だが，途中で低くなる．i番目のイベントのみ標準形で，k番目のイベントはオリジナルのままで分類する
- $LH(1)$　：$P_{k,j} < P_{i,1}$かつ$P_{k,j+1} < P_{i,1}$となるjが存在する．つまり，途中から$P_{i,1}$よりプライオリティが高くなる場合である
- $L(1)$　：すべてのプライオリティ($P_{k,1}$，$P_{k,2}$，$P_{k,3}$，…)が$P_{i,1}$より低い

〔図11〕解析手順の概要

〔図12〕標準形への変換

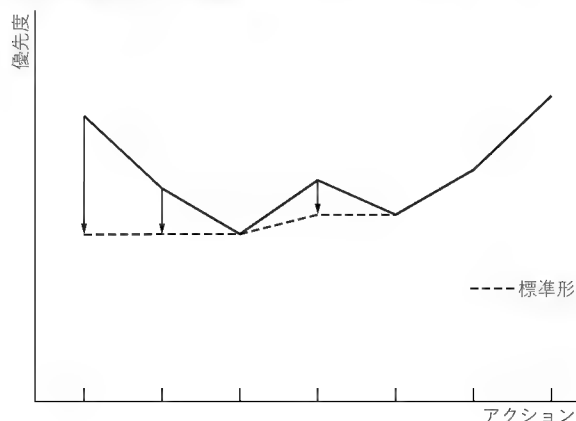
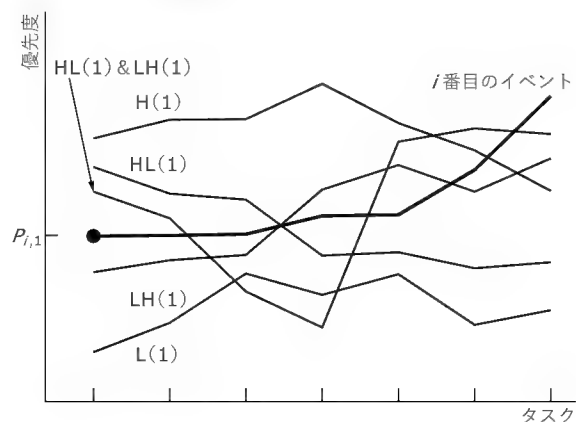
〔図13〕イベントの分類

この分類は，いわゆる同値関係から導かれる類別を作っているわけではないので．一つのイベントが$HL(1)$でありかつ$LH(1)$である場合もあり得る．

たとえば，イベント3に対する分類は次のようになる(**図13**)．

- H(1)　：イベント1，イベント2
- HL(1)：イベント6
- LH(1)：イベント4
- L(1)　：イベント5

▶ Step3　$P_{i,1}$に関するビジィ期間（busy period）を計算する

ビジィ期間とは，要求された仕事が終わるまでの期間である．最悪のタイミングで仕事を要求されたときのビジィ期間が重要で，それが最悪応答時間になる．Step3では，各イベントの第1アクションのビジィ期間を求める．最悪のタイミングは，一般にはすべてのイベントが同時に入った場合である．しかし，プライオリティが変化する場合やメッセージがキューイングされる場合は，個別に検討しなければならない場合がある(**図14**)．

たとえばイベント3の場合では，第1アクション(task_A4)のプライオリティが180なので，イベント3が入力されても，プライオリティ180以上のアクションがあれば，そちらが先に実行される．そして，その後task_A4が実行される．task_A4が終了するまでの時間を求めるのがStep3である．この期間中にも入力周期の短いイベントの入力があり得るので話が複雑になる．また，最悪のタイミングも時刻0で全イベントが入力されるよりは，イベント4の入力が先にあり，イベント4の第1アクション(task_A5)が終了して第2アクションの実行が始まると同時に，他のイベントの入力があった場合がイベント3にとっては最悪になる．それで，この最悪の初期状態から出発して反復計算によりビジィ期間を求めることになる．計算手順を一般化すると次のようになる．

イベントiについて，

$$S = B_i + \underset{k\in LH(1)}{MAX}\left[HSeg_k(1)\right] + \sum_{k\in HL(1)} FSeg_k(1)$$

$$a_0 = S + C_i + \sum_{k\in H(1)} C_k$$

によって初期値を求め，以後

$$a_{n+1} = S + \left\lceil \frac{a_n}{T_i} \right\rceil C_i + \sum_{k\in H(1)} \left\lceil \frac{a_n}{T_k} \right\rceil C_k^i$$

による計算を$a_{n+1} = a_n$になるまで繰り返す[注1]．ここでB_iは，クリティカルセクションなどによってイベントiのアクション列がブロックされる時間である．ここで扱っている例では0である．$HSeg_k(1)$は，$P_{i,1}$より高プライオリティのアクションが続く部分(H-Segment)である．その中で最長のものを一つSに加える．イベント3の場合の，イベント4の第2・第3アクションの実行

注1：⌈ ⌉は切り上げ記号である．

〔図14〕各アクションのプライオリティ

イベント3に対する分類

イベント	type	action 1	action 2	action 3	action 4	action 5
1	H(1)	200	201			
2	H(1)	190	190	202	190	
3	—	180				
4	LH(1)	150	201	202	150	150
5	L(1)	140	140			
6	HL(1)	201	202	120		

task_A4よりもプライオリティの高いタスク□が先に実行される

時間である．イベント3では$LH(1)$に分類されるものがイベント4だけだが，複数ある場合はその最大値を使う．$LH(1)$に分類されたイベントは前後を$P_{i,1}$よりも低いプライオリティのタスクにはさまれているので，1回だけ評価すればよいのである．ただし，$LH(1)$に分類されるイベントが複数ある場合には，それぞれの$FSeg_k(1)$の最大値を加えなければならない．

ハンドブックに載っている計算手順は簡略化されている．したがって正確には，

$$= B_i + \sum_{k \in LH(1)} MAX\left[HSeg_k(1)\right] + \sum_{k \in HL(1)} FSeg_k(1)$$

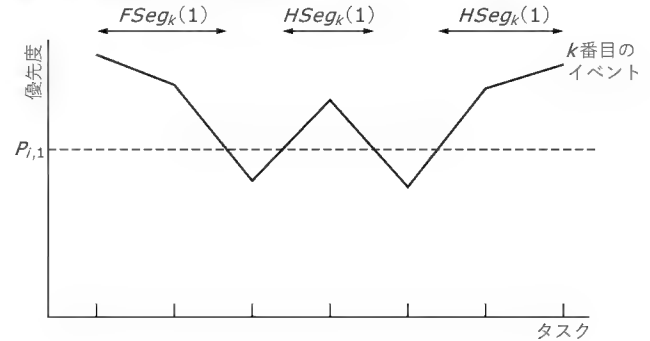
とすべきである．同一のイベントの中に複数の$HSeg_k(1)$がある場合には，その中から最長のものを選ぶ，それを$LH(1)$イベントについて繰り返さなければならない．$FSeg_k(1)$は第1アクションを含んだH-Segmentである．ハンドブックでは「F-Segment」と呼んでいる．一つのイベントが$LH(1)$と$HL(1)$に同時に含まれる場合には，$FSeg_k(1)$と$MAX[HSeg_k(1)]$の長いほうを選択しなければならない．この部分もハンドブックでは省略されている．さらにまた，**図15**に示したように，最後の$HSeg_k(1)$と$FSeg_k(1)$が連結する場合などもあり得るので注意が必要である．

このあたりはオリジナルの論文を見たほうが正確である．基本は，$P_{i,1}$よりも低いプライオリティを含んでいるタスク列は一度だけ評価すればよいということである．デッドラインが起動周期よりも長い場合は，キューイングされるので$FSeg_k(1)$の部分を複数評価する必要があるが，ここでは扱っていない．

たとえばイベント3について計算すると，次のようになる．

$$\begin{aligned} S &= B_3 + HSeg_4(1) + FSeg_6(1) \\ &= 0 + 21 + 21 \\ &= 42 \\ a_0 &= S + C_3 + C_1 + C_2 \\ &= 42 + 9 + 13 + 34 \\ &= 98 \\ a_1 &= 42 + \left\lceil \frac{98}{160} \right\rceil 9 + \left\lceil \frac{98}{100} \right\rceil 13 + \left\lceil \frac{98}{100} \right\rceil 34 \\ &= 98 = a_0 \end{aligned}$$

〔図15〕$FSeg_k(1)$と$HSeg_k(1)$

となる．

ビジィ期間が求まったら，その間に起動される回数を計算する．この場合ビジィ期間が98なので，起動周期160のイベント3は1回だけ起動されることになる．イベント5について計算するとビジィ期間は338となり，起動周期は325なのでこの間にイベントが2回入ることになる．スケジューリングの世界では，起動されたタスク一つ一つを指す用語としてジョブ(Job)を使用する．Step3で，応答時間を調べる必要のあるイベントごとのジョブの数がわかる．この数をイベントごとにNとする．

イベント3の場合，アクション数は1でジョブ数が1だったので，ここで求めた98が最悪応答時間になる．すなわち，Step4以降はイベント3については必要ない．98はデッドライン160より短いので，イベント3についてはスケジュール可能となる．

▶ Step4　第1アクションの終了時間の計算

ジョブの回数をbとして，そのb番目のジョブにおける第1アクションの終了時間は，Step3とほとんど同一の式によって計算できる．

$$S = B_i + C_{i,1} + (b-1)C_1 + \underset{k \in LH(1)}{MAX}\left[HSeg_k(1)\right] + \sum_{k \in HL(1)} FSeg_k(1)$$

$$a_0 = S + \sum_{k \in H(1)} C_k$$

を初期値として，

$$a_{n+1} = S + \sum_{k \in H(1)} \left\lceil \frac{a_n}{T_k} \right\rceil C_k$$

を，Step3と同様に$a_{n+1} = a_n$になるまで繰り返す．そのときのa_{n+1}が終了時間となる．上記のS式もハンドブックでは簡略化されているので，注意が必要である．

▶ Step5　第jアクションのプライオリティに対するイベントの再分類

最初にStep5を実行する際には，第2アクションに対して分類することになる．

ここで考慮しなければならないイベントは，Step2で$H(1)$に分類されたイベントのみである．応答時間を求めようとしているイベントについては標準形を使用しているので，アクションが進むにしたがってプライオリティが高くなる．高くなると，い

ままで$H(1)$に分類されていたイベントが$HL(2)$になる可能性があるので，それを確認する．

アクション数をjとすると，$H(j-1)$に分類されていたイベントは$H(j)$と$HL(j)$に分割される．$H(j)$は，すべてのアクションのプライオリティが$P_{i,j}$以上のもので，$HL(j)$はそれ以外である．

$HL(j+1)$に分類されたイベントは，$FSeg_k(j)$を1回だけ評価すれば十分である．

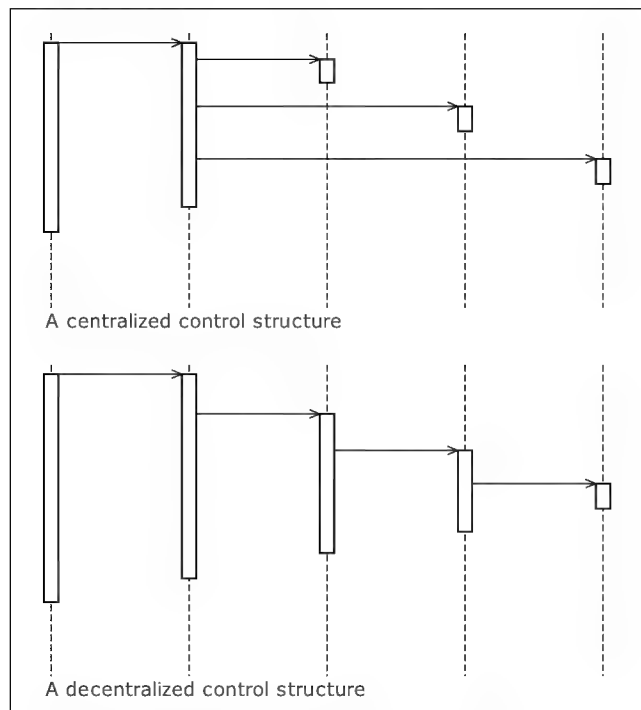
▶ Step 6　第bジョブの第jアクションの終了時間の計算

Step5の分類にしたがってj番目のアクションの終了時間を計算する．計算方法は，いままでと同様の繰り返し計算である．

〔図16〕各実装の応答時間

イベント	deadline	response time		
		個別タスク	サーバタスク	リエントラント関数
1	100	55	22	13
2	130	89	59	47
3	160	98	68	56
4	225	187	178	123
5	325	338	219	194
6	380	372	372	372

デッドラインオーバ

〔図17〕制御の型

A centralized control structure in a flow of events produces a "fork-shaped" sequence diagram. A decentralized control structure produces a "stairway-shaped" sequence diagram

Rational Unified Process, Guidelines: Sequence Diagram より

$$S = S' + C_{i,j} + \sum_{k \in HL(j)} FSeg_k(j) + \sum_{k \in H(j)} \left\lceil \frac{a_n}{T_k} \right\rceil C_k$$

ここでS 'は，イベントiに対して以前使用したSの値である．a_nはStep4または第$j-1$アクションについてStep6で求めた値である．

これらを初期値として，

$$a_{n+1} = S + \sum_{k \in H(j)} \left\lceil \frac{a_n}{T_k} \right\rceil C_k$$

を$a_{n+1}=a_n$になるまで繰り返す．計算が収束したらStep5に戻って，次のアクションについて繰り返す．すべてのアクションが終了したら，そのときのa_nをb番目のジョブの応答時間$E(b)$とする．そして，$b=b+1$としてStep4に戻る．このときのbがStep2で求めたジョブ数Nを超えていれば，Step7に進む．

▶ Step7　最悪応答時間の選択

N回のジョブの中でのもっとも長い応答時間を求め，それをイベントiの最悪応答時間R_iとする．

$$R_i = \underset{1 \le b \le N}{MAX}\left[E(b) - (b-1)T_i\right]$$

以上の手順によって計算した各イベントに対する応答時間は，**図16**の個別タスクの欄のようになる．解析の結果，イベント5がデッドラインオーバしてしまうことがわかる．

● サーバタスクによる実装

次の実装方式は，サーバタスクとイベントタスクを使用する方法である．シーケンス図で表す代表的な制御パターンには，**図17**で示す2とおりがある．個別タスクによる方法は，**図17**の下側にあたる．サーバタスクによる実装は上側に対応する．

この実装では，いままで個別タスクといっていたアクションを処理するタスクをサーバタスクとする．いままでと違うのは，サーバタスクは次の処理が何になるのか知る必要がない点である．アクション順は，新たに作るイベントタスクで実装する．機能の実装とそれをまとめる動作メカニズムを別々に実装するアーキテクチャである．タスク数が増えるので，古参の組み込みエンジニアが嫌がる設計かもしれない．サーバタスクとイベントタスクのスケルトンは，**図18**のようになる．また全体のタスク構成は，**図19**のようになる．

今回の構成で変化したのは，タスク数だけではなく共有サーバタスクのプライオリティである．task_AS2はtask_AS1から直接呼び出されないので，プライオリティが202から191に下がっている．タイミング解析的にはここがポイントである．応答時間の計算法は個別タスクの場合とまったく同様である．計算結果はまとめて**図19**に示した．この計算では，イベントタスクの実行時間は無視している．この扱いはあまり厳密ではないかもしれないが，メッセージを送るだけのイベントタスクの実行時間はアクション処理時間と比較して無視できる程度とすることは妥当だと思う．計算結果に対して考察を加えるとすれば，比較的実行時間の長いtask_AS2のプライオリティを下げたのでCPU利用効率が向上し，応答時間が改善されデッドラインオー

バがなくなったと考えられる．

● 関数による実装

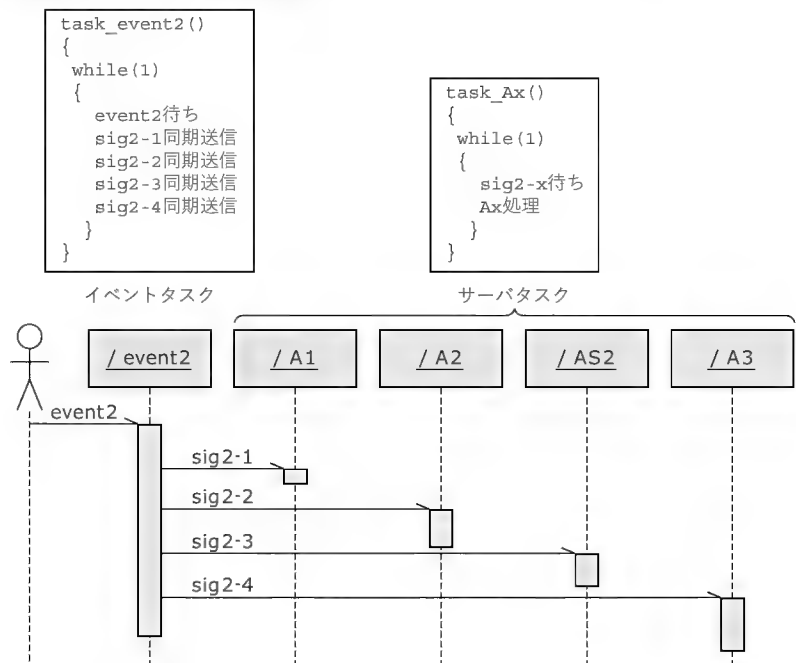
最後の実装方法は，それぞれのアクションをリエントラントな関数で実現するものである．イベントタスクがアクション関数を呼び出すアーキテクチャになる．じつは，この方法がいちばん効率の良い設計である．また，タイミング解析も簡単である[3]．それぞれのアクション関数の実行時間の和をタスクの実行時間として計算すればよい．ただしリエントラントにしなければならないので，いつでも利用できるとはかぎらない．リエントラントにできない場合は，クリティカルセクションを設けることになる．

3種類の実装方法と応答時間をハンドブックにしたがって紹介したが，実際の設計ではこれらを混ぜ合わせて使用することになる．アクションを役割ととらえてオブジェクトにより実装するオブジェクト指向は，タスク分割を先に行ってしまう設計方法に比べて柔軟性があるように思う．

〔図18〕サーバタスクによる実装

おわりに

今回扱った話題は二つあって，一つはプライオリティが変化するタスク列のスケジュール判定法，もう一つは実装方法による効率の違いである．

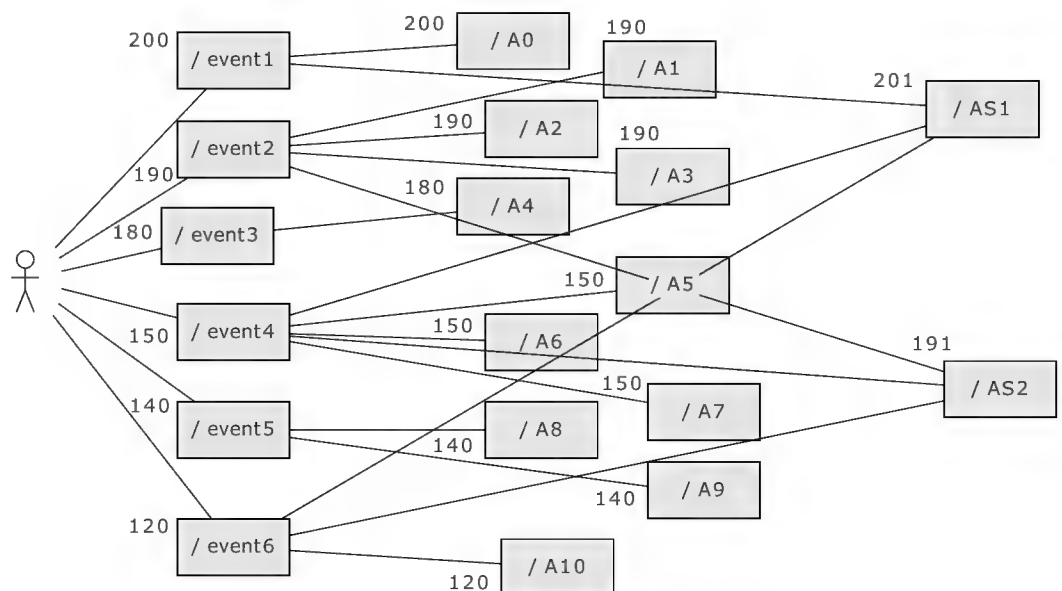
タスク列による設計仕様はシーケンス図と相性が良いので，最近はよく出てくる．しかしそのわりには，どのようにタイミング解析するのかについてはあまり知られていないように思う．タイミング解析することでタスクを増やすことが，必ずしも応答時間を長くすることにはならないこと等がわかってくるのである．定量的な設計をすることで，一般論から離れてアプリケーションに合った的確な設計が可能になる．

参考文献

1) Mark H. Klein ほか，*A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems*（The Kluwer International Series in Enginee），Kluwer Academic Pub，ISBN: 0792393619，1993/11/01

2) Michael G. Harbour，"Fixed priority scheduling of periodic tasks with varying execution priority"，*Proceedings of the IEEE Real-Time Systems Symposium*, pp.116-128, 1991

3) 藤倉俊幸，『リアルタイム/マルチタスクシステムの徹底研究』，TECH-I vol.15，第18章，CQ出版(株)

ふじくら・としゆき　IBM SWG/ラショナルソフトウェア

〔図19〕サーバタスクによる実装(その2)

第10回

続・C99規格についての説明と検証

岸　哲夫

前回に引き続き，C99規格についての説明と検証について解説する．今回は，
- 追加された関数やマクロについて
- 新規に追加された標準ライブラリについて
- C99規格対応に関するGNU Cのバージョンごとの進捗

に関して，説明と検証を行う．

（筆者）

今回も，前回に引き続き「ISO/IEC 9899：1999 - Programming Language C」(略称：C99)規格について説明と検証をします．

追加された関数やマクロについて

● マクロDECIMAL_DIG

これについては，完全にフィックスされていないのが現状です．GCC3.2から導入されています．次のように定義されています．

```
# define DECIMAL_DIG      21
```

● マクロFLT_EVAL_METHOD

GCC3.2から導入されています．以下のように定義されています．

```
# define FLT_EVAL_METHOD 2
```

〔リスト1〕va_copyの例(test126.c)

```
#include <stdarg.h>
/*
 *va_copyについて
 */
#include <stdio.h>
void func(int num, ...);
main()
{
    func(4,"aa","bb","cc","dd");
    func(6,"aa-","bb-","cc-","dd-","ee-","ff-");
}
void func(int num, ...)
{
    va_list ap;
    va_list dup_ap;
    char *str;
    int  ix;
    va_start(ap, num);
    __va_copy(dup_ap,ap);
    for(ix=0;ix<num;ix++)
    {
         printf("apの%d番目の引き数は%s¥n",ix+1,va_arg(ap, char *));
    }
    printf("¥n");
    for(ix=0;ix<num;ix++)
    {
         printf("dup_apの%d番目の引き数は%s¥n",ix+1,
                                          va_arg(dup_ap, char *));
    }
    printf("¥n");
    va_end(ap);
    va_end(dup_ap);
}
```

実際にこの値が意味することは，以下のとおりです．

- FLT_EVAL_METHODの値が－1：浮動小数点計算の方式は不定である
- FLT_EVAL_METHODの値が0：型を保持して計算する
- FLT_EVAL_METHODの値が1：floatはDoubleに型拡張される．double，long doubleはそのまま
- FLT_EVAL_METHODの値が2：float，doubleはlong doubleに型拡張される．long doubleはそのまま

つまり，GCC3.2では「float，doubleはlong doubleに型拡張され，long doubleはそのまま」という計算方式になります．これからGCC3.2の導入を考えている方は，心にとどめておいてください．

● マクロva_copy

リスト1に使用例を，**リスト2**にマクロ展開リストの一部を，**リスト3**に生成されたアセンブラコードを示します．以下は実行結果です．

```
$ ./test126
apの1番目の引き数はaa
apの2番目の引き数はbb
apの3番目の引き数はcc
apの4番目の引き数はdd

dup_apの1番目の引き数はaa
dup_apの2番目の引き数はbb
dup_apの3番目の引き数はcc
dup_apの4番目の引き数はdd

apの1番目の引き数はaa-
apの2番目の引き数はbb-
apの3番目の引き数はcc-
apの4番目の引き数はdd-
apの5番目の引き数はee-
apの6番目の引き数はff-
```

〔リスト2〕マクロ展開リストの一部(test126.src)

```
void func(int num, ...)
{
    va_list ap;
    va_list dup_ap;
    char *str;
    int  ix;
    ( ap  = ((__gnuc_va_list) __builtin_next_arg (  num ))) ;
    ( dup_ap ) = ( ap ) ;
    for(ix=0;ix<num;ix++)
    {
        printf("apの%d番目の引き数は%s¥n",ix+1,( ap  = (__gnuc_va_list) ((char *) ( ap ) + (((sizeof (   char *  ) + sizeof (int) - 1)
                        / sizeof (int)) * sizeof (int)) ),  *((  char *  *) (void *) ((char *) ( ap ) -
                        (((sizeof (   char *  ) + sizeof (int) - 1) / sizeof (int)) * sizeof (int)) ))) );
    }
    printf("¥n");
    for(ix=0;ix<num;ix++)
    {
        printf("dup_apの%d番目の引き数は%s¥n",ix+1,( dup_ap  = (__gnuc_va_list) ((char *) ( dup_ap ) + (((sizeof (   char *  ) +
                        sizeof (int) - 1) / sizeof (int)) * sizeof (int)) ),  *((  char *  *) (void *) ((char *) ( dup_ap ) -
                        (((sizeof (   char *  ) + sizeof (int) - 1) / sizeof (int)) * sizeof (int)) ))) );
    }
    printf("¥n");
    ((void)0) ;
    ((void)0) ;
}
```

〔リスト3〕生成されたアセンブラコード(test126.s)

```
    .file     "test126.c"
    .version  "01.01"
gcc2_compiled.:
.section .rodata
.LC0:
    .string   "dd"
.LC1:
    .string   "cc"
.LC2:
    .string   "bb"
.LC3:
    .string   "aa"
.LC4:
    .string   "ff-"
.LC5:
    .string   "ee-"
.LC6:
    .string   "dd-"
.LC7:
    .string   "cc-"
.LC8:
    .string   "bb-"
.LC9:
    .string   "aa-"
.text
    .align 4
.globl main
    .type      main,@function
main:
    pushl %ebp
    movl %esp,%ebp
    subl $8,%esp
    addl $-12,%esp
    pushl $.LC0
    pushl $.LC1
    pushl $.LC2
    pushl $.LC3
    pushl $4
    call func
    addl $32,%esp
    addl $-4,%esp
    pushl $.LC4
    pushl $.LC5
    pushl $.LC6
    pushl $.LC7
    pushl $.LC8
    pushl $.LC9
    pushl $6
    call func
    addl $32,%esp
.L2:
    movl %ebp,%esp
    popl %ebp
    ret
.Lfe1:
    .size      main,.Lfe1-main
.section .rodata
.LC10:
    .string   "ap¥244¥316%d¥310¥326¥314¥334
      ¥244¥316¥260¥372¥277¥364¥244¥317%s¥n"
.LC11:
    .string   "¥n"
.LC12:
    .string   "dup_ap¥244¥316%d¥310¥326¥314
  ¥334¥244¥316¥260¥372¥277¥364¥244¥317%s¥n"
.text
    .align 4
.globl func
    .type      func,@function
func:
    pushl %ebp
    movl %esp,%ebp
    subl $24,%esp
    leal 12(%ebp),%eax
    movl %eax,-4(%ebp)
    movl -4(%ebp),%eax
    movl %eax,-8(%ebp)
    movl $0,-16(%ebp)
    .p2align 4,,7
.L4:
    movl -16(%ebp),%eax
    cmpl 8(%ebp),%eax
    jl .L7
    jmp .L5
    .p2align 4,,7
.L7:
    addl $-4,%esp
    addl $4,-4(%ebp)
    movl -4(%ebp),%eax
    addl $-4,%eax
    movl (%eax),%edx
    pushl %edx
    movl -16(%ebp),%eax
    incl %eax
    pushl %eax
    pushl $.LC10
    call printf
    addl $16,%esp
.L6:
    incl -16(%ebp)
    jmp .L4
    .p2align 4,,7
.L5:
    addl $-12,%esp
    pushl $.LC11
    call printf
    addl $16,%esp
    movl $0,-16(%ebp)
    .p2align 4,,7
.L8:
    movl -16(%ebp),%eax
    cmpl 8(%ebp),%eax
    jl .L11
    jmp .L9
    .p2align 4,,7
.L11:
    addl $-4,%esp
    addl $4,-8(%ebp)
    movl -8(%ebp),%eax
    addl $-4,%eax
    movl (%eax),%edx
    pushl %edx
    movl -16(%ebp),%eax
    incl %eax
    pushl %eax
    pushl $.LC12
    call printf
    addl $16,%esp
.L10:
    incl -16(%ebp)
    jmp .L8
    .p2align 4,,7
.L9:
    addl $-12,%esp
    pushl $.LC11
    call printf
    addl $16,%esp
.L3:
    movl %ebp,%esp
    popl %ebp
    ret
.Lfe2:
    .size      func,.Lfe2-func
    .ident    "GCC: (GNU) 2.95.3
                        20010315 (release)"
```

dup_apの１番目の引き数はaa-
dup_apの２番目の引き数はbb-
dup_apの３番目の引き数はcc-
dup_apの４番目の引き数はdd-
dup_apの５番目の引き数はee-
dup_apの６番目の引き数はff-

このようにva_startや__va_copyはマクロになっています．va_endは，ここで見るかぎり不要に見えますが，処理によっては必要になるはずなので，va_startや__va_copyごとに必ず定義してください．

アセンブラソースを見てもわかるように，__va_copyマクロはアドレスをコピーするだけではなく，領域を確保してコピーを行っています．よって，元のデータを破壊してもコピー先は保持されます．

math.hのマクロ

C99規格ではmath.hにもマクロや関数が追加されています．

〔リスト4〕**HUGE_VALの例**(test127.c)

```
/*
 *HUGE_VALについて
 */
#include <stdio.h>
#include <math.h>
main()
{
    double  a  =  HUGE_VAL;
    printf("%G¥n",a);
}
```

〔リスト5〕**生成されたアセンブラコード**(test127.s)

```
    .file    "test127.c"
    .def ___main; .scl 2;  .type    32;  .endef
    .text
LC0:
    .ascii "%G¥12¥0"
    .align 2
.globl _main
    .def _main;   .scl 2;  .type    32;  .endef
_main:
    pushl    %ebp
    movl %esp, %ebp
    subl $40, %esp
    andl $-16, %esp
    movl $0, %eax
    movl %eax, -12(%ebp)
    movl -12(%ebp), %eax
    call __alloca
    call ___main
    movl __imp____infinity, %eax
    movl 4(%eax), %edx
    movl (%eax), %eax
    movl %eax, -8(%ebp)
    movl %edx, -4(%ebp)
    movl $LC0, (%esp)
    movl -8(%ebp), %eax
    movl -4(%ebp), %edx
    movl %eax, 4(%esp)
    movl %edx, 8(%esp)
    call _printf
    leave
    ret
    .def _printf;  .scl 2;  .type    32;  .endef
```

〔リスト6〕**INFINITYの例**(test128.c)

```
/*
 *INFINITYについて
 */
#include <stdio.h>
#include <math.h>
main()
{
    float   f   =   NAN;
    printf("%f¥n",f);
}
```

しかし，現在の評価環境であるGCC2.95.3では対応できていません．C99規格の説明をするために，しばらくの間Cygwin環境のお世話になることにします．GCCのバージョンは3.2です．

● HUGE_VALについて

GCC2.95でもHUGE_VALは定義されていました．

今回HUGE_VALF/HUGE_VALLも規格に追加されたのですが，GCC3.2の環境でもまだ対応できていません．

ちなみに，HUGE_VALはdouble型の無限大，HUGE_VALFはfloat型，HUGE_VALLはlong doubleの無限大を表す定数です(**リスト4**，**リスト5**)．

以下は実行結果です．

```
$ ./test127
Inf
```

このように，無限大を表すInfが表示されます．

● INFINITY

無限大を表すfloat型の定数ですが，GCC3.2でも定義されていません(**リスト6**)．コンパイル結果は以下のとおりです．

```
$ gcc test128.c -o test128
test128.c: In function `main':
test128.c:8: `INFINITY' undeclared
                (first use in this function)
test128.c:8:(Each undeclared identifier is
                              reported only once
test128.c:8: for each function it
                                     appears in.)
```

● NAN

not a numberの意味です．0.0/0.0の値をNANと名付けますが，GCC3.2の環境でも定義されていません．

● fpclassify(x)

これは関数型マクロです．引き数xがNANか無限大，または通常の数値，ゼロかを判定し，結果を，それぞれFP_NAN，FP_INFINITE，FP_NAN，FP_NORMAL，FP_ZEROとして返します．FP_SUBNORMALという定数もあるはずですが，GCC3.2では存在しません(**リスト7**)．

実行結果は以下のとおりです．

```
$ ./test129
0.0/0.0 = not a number
1.0/0.0 = 無限大
1.1/1.5 = 通常の数値
```

● isfinite(x)

この関数型マクロは，引き数が無限大かNANの場合にゼロを返します(**リスト8**)．実行結果は以下のとおりです．

```
$ ./test130
isfinite(0.0/0.0) は 真
```

● isinf(x)

この関数型マクロは，引き数が無限大の場合にゼロを返します(**リスト9**)．実行結果は次のとおりです．

〔リスト7〕**fpclassifyの例**(test129.c)

```
/*
 *fpclassifyについて
 */
#include <stdio.h>
#include <math.h>
main()
{
    if  (   fpclassify(0.0/0.0)==   FP_NAN    )
    {
        printf("0.0/0.0 = not a number¥n");
    }
    if  (   fpclassify(1.0/0.0)==   FP_INFINITE   )
    {
        printf("1.0/0.0    = 無限大¥n");
    }
    if  (   fpclassify(1.1/1.5)==   FP_NORMAL )
    {
        printf("1.1/1.5    = 通常の数値¥n");
    }
}
```

〔リスト8〕**isfiniteの例**(test130.c)

```
/*
 *isfiniteについて
 */
#include <stdio.h>
#include <math.h>
main()
{
    if  (   isfinite(0.0/0.0)   )
    {
        printf("isfinite(0.0/0.0) は 真");
    }
    if  (   isfinite(1.0/0.0)   )
    {
        printf("isfinite(1.0/0.0) は 真");
    }
    if  (   isfinite(1.1/1.5)   )
    {
        printf("isfinite(1.1/1.5) は 真");
    }
}
```

〔リスト9〕**isinfの例**(test131.c)

```
/*
 *isintについて
 */
#include <stdio.h>
#include <math.h>
main()
{
    if  (   isinf(0.0/0.0) )
    {
        printf("isinf(0.0/0.0) は 真");
    }
    if  (   isinf(1.0/0.0) )
    {
        printf("isinf(1.0/0.0) は 真");
    }
    if  (   isinf(1.1/1.5) )
    {
        printf("isinf(1.1/1.5) は 真");
    }
}
```

〔リスト10〕**isnanの例**(test132.c)

```
/*
 *isnanについて
 */
#include <stdio.h>
#include <math.h>
main()
{
    if  (   isnan(0.0/0.0) )
    {
        printf("isnan(0.0/0.0) は 真");
    }
    if  (   isnan(1.0/0.0) )
    {
        printf("isnan(1.0/0.0) は 真");
    }
    if  (   isnan(1.1/1.5) )
    {
        printf("isnan(1.1/1.5) は 真");
    }
}
```

```
$ ./test131
isinf(1.0/0.0) は 真
```

● isnan(x)

この関数型マクロは，引き数がNANの場合にゼロを返します(**リスト10**)．実行結果は以下のとおりです．

```
$ ./test132
isnan(0.0/0.0) は 真
```

以上のように，math.hで定義されている関数の種類が増えました．

それはfloat，long doubleに対応するようになったからです．同一の関数でもdouble，float，long double対応と3種類あることが多くなりました．

しかし，GCC3.2ではlong doubleには対応できていません．詳しくはmath.hをご覧ください．

stdio.hに追加されたいくつかの関数

● snprintf

この関数の使用例を**リスト11**に，実行結果を**リスト12**に示します．編集して書き出す場合に便利に利用できます．sprintfだけでは，転送元のデータがどのくらいの大きさか調べないと使いにくかったのですが，これで万全です．

● vsnprintf

この関数の使用例を**リスト13**に，実行結果を**リスト14**と**リスト15**(p.137)に示します．可変引き数リストを使用したい場合，このvsnprintfを使えば簡単にできるはずです．

● vsscanf

入力データ中のaa--bb--cc--dd--というデータが分割されて可変引き数リストに読み込まれました．同じようにvscanfやvfscanfも利用できます．使用例を**リスト16**(p.137)に，実行結果を**リスト17**(p.137)に示します．

stdlibに追加されたいくつかの関数

● strtof，strtold関数

GCC3.2の環境ではstrtoldが定義されていません．strtofは問題なく動作します(**リスト18**，p.137)．実行結果は次のとおりです．

〔リスト 11〕snprintf の例(test134.c)

```
/*
 *snprintfについて
 */
#include <stdio.h>
#include <math.h>
main()
{
    FILE *out;
    char buf[1024];
    snprintf(buf,10,"12345678901234567890");
    printf("%s¥n",buf);
    snprintf(buf,50,"12345678901234567890");
    printf("%s¥n",buf);
    out  =    fopen("test.dat","w");
    fwrite(buf,1,1024,out);
    fclose(out);
}
```

〔リスト 13〕vsnprintf の例(test135.c)

```
#include <stdarg.h>
/*
 *vsnprintfについて
 */
#include <stdio.h>
void func(int num, ...);
main()
{
    func(4,"aa","bb","cc","dd");
}
void func(int num, ...)
{
    FILE *out1;
    FILE *out2;
    va_list ap;
    va_list dup_ap;
    char *str;
    char buf[1024];
    int  ix;
    va_start(ap, num);
    __va_copy(dup_ap,ap);
    vsnprintf(buf,10,"%s:%s:%s:%s:%s",ap);
    out1 =    fopen("test1.dat","w");
    fwrite(buf,1,1024,out1);
    fclose(out1);
    vsnprintf(buf,20,"%s--%s--%s--%s--%s--%s",dup_ap);
    out2 =    fopen("test2.dat","w");
    fwrite(buf,1,1024,out2);
    fclose(out2);
    va_end(ap);
    va_end(dup_ap);
}
```

〔リスト 12〕リスト 11 の実行結果データダンプリスト

```
  Addr     0 1  2 3  4 5  6 7  8 9  A B  C D  E F 0 2 4 6 8 A C E
--------  ---- ---- ---- ---- ---- ---- ---- ---- ----------------
00000000  3132 3334 3536 3738 3930 3132 3334 3536 1234567890123456
00000010  3738 3930 0001 0000 3cfb 2200 c876 f877 7890....<{".Hvxw
00000020  aafb 2200 3502 0000 0000 d877 3cc1 dc77 *{".5.....Xw<A¥w
00000030  10ca dc77 a8fb 2200 0100 0000 0000 0000 .J¥w({".........
00000040  c800 d877 0cfb 2200 10ca dc77 40b8 dc77 H.Xw.{"..J¥w@8¥w
00000050  0000 0000 ac01 0000 60fb 2200 0000 0000 ....,...`{".....
00000060  e538 f877 0000 2300 08b6 2300 0000 0000 e8xw..#..6#.....
00000070  3cfb 2200 0000 0000 3800 0000 2000 0000 <{".....8... ...
00000080  2000 0000 6cfb 2200 f8eb fd7f 581b 0d61  ...l{".xk}.X..a
00000090  3400 00c0 28fc 2200 2053 0061 3400 00c0 4..@(|". S.a4..@
000000a0  50b6 e577 1800 0000 0030 0000 fcff ab00 P6ew.....0..|.+.
000000b0  2000 0000 f8eb fd7f 0000 0000 f8eb fd7f  ...xk}.....xk}.
000000c0  0000 0000 f8eb fd7f e800 0000 0099 0d61 ....xk}.h......a
000000d0  0300 0000 f8eb fd7f 0000 0000 7900 2300 ....xk}.....y.#.
000000e0  00e0 fd7f b57c f877 e8fb 2200 1800 0000 .`}.5|xwh{".....
000000f0  e400 0000 0000 2300 0300 0000 0000 0000 d.....#.........
                          ～以下略～
```

〔リスト 14〕リスト 13 の実行結果データダンプリスト(その 1)

```
  Addr     0 1  2 3  4 5  6 7  8 9  A B  C D  E F 0 2 4 6 8 A C E
--------  ---- ---- ---- ---- ---- ---- ---- ---- ----------------
00000000  6161 3a62 623a 6363 3a00 0100 80b4 e577 aa:bb:cc:....4ew
00000010  2c00 0000 d803 5f61 581b 0d61 4fd8 d4ac ,...X._aX..aOXT,
00000020  0700 0100 0000 0000 f8fa 2200 0000 0000 ........xz".....
00000030  e538 f877 0000 2300 68c7 2300 0000 0000 e8xw..#.hG#.....
00000040  10ca dc77 40b8 dc77 0000 0000 8a01 0000 .J¥w@8¥w........
00000050  3cfb 2200 c876 f877 aafb 2200 3502 0000 <{".Hvxw*{".5...
00000060  0000 d877 3cc1 dc77 10ca dc77 a8fb 2200 ..Xw<A¥w.J¥w({".
00000070  0100 0000 0000 0000 c800 d877 0cfb 2200 ........H.Xw.{".
00000080  10ca dc77 58fb 2200 0b00 0000 7039 f877 .J¥wX{".....p9xw
00000090  0000 2300 9808 2300 0b00 0000 0000 0000 ..#...#.........
000000a0  30fb 2200 0002 0000 0000 2300 10cb 2300 0{".......#..K#.
000000b0  7801 2300 9e00 0000 50b6 fc77 16b5 fc77 x.#.....P6|w.5|w
000000c0  2db5 fc77 00ec fd7f f8eb fd7f 0000 0000 -5|w.l}.xk}.....
000000d0  0000 2300 b8ca 2300 7801 2300 a900 0000 ..#.8J#.x.#.)...
000000e0  50b6 fc77 16b5 fc77 2db5 fc77 7801 2300 P6|w.5|w-5|wx.#.
000000f0  7801 2300 442b 4000 0000 0000 4006 2300 x.#.D+@.....@.#.

  Addr     0 1  2 3  4 5  6 7  8 9  A B  C D  E F 0 2 4 6 8 A C E
--------  ---- ---- ---- ---- ---- ---- ---- ---- ----------------
00000100  e800 0000 0099 0d61 0300 0000 f8eb fd7f h......a....xk}.
00000110  0000 0000 7801 2300 7801 2300 b57c f877 ....x.#.x.#.5|xw
00000120  e8fb 2200 4006 2300 e400 0000 0000 2300 h{".@.#.d.....#.
00000130  0300 0000 0000 0000 0000 0000 0000 0000 ................
                          ～中略～
000003a0  1030 4000 46f0 0ddf 68fe 2200 f6b4 0861 .0@.Fp._h~".v4.a
000003b0  f803 040a 60fe 2200 68fe 2200 5014 4000 x...`~".h~".P.@.
000003c0  2000 040a 442b 4000 88fe 2200 9814 4000  ...D+@..~"...@.
000003d0  e003 040a 0050 4000 0000 0000 0000 0000 `....P@.........
000003e0  402b 4000 a02a 4000 98fe 2200 5f10 4000 @+@. *@..~"._.@.
000003f0  0040 4000 0050 4000 b8fe 2200 915d 0061 .@@..P@.8~"..].a
```

```
$ ./test137
0.123655
```

● strtoll，strtoull 関数

どちらも問題なく動作しました(**リスト 19**，p.138)．実行結果は以下のとおりです．

```
$ ./test138
-965485412
98612464
```

● atoll，llabs，lldiv 関数

GCC3.2 の環境において，この中では `llabs` だけ定義されていました(**リスト 20**，p.138)．実行結果は以下のとおりです．

```
$ ./test139
653587458
```

C99 規格で新規に追加された標準ライブラリ

● 複素数について

これはまだ対応できていないようです．対応されると複素数の扱いが楽になるので，マンデルブロート集合の画像が作りやすくなるでしょう．

● 浮動小数点について

これもまだ対応できていません．これは各環境で独自に扱っていた浮動小数点の扱い方を統一したものです．しかし，これに対応するということは，いままでの方式を変更しなければならないので，各環境で問題が残るでしょう．

〔リスト15〕リスト13の実行結果データダンプリスト(その2)

```
  Addr     0 1  2 3  4 5  6 7  8 9  A B  C D  E F 0 2 4 6 8 A C E
--------  ---- ---- ---- ---- ---- ---- ---- ---- ----------------
00000000  6161 2d2d 6262 2d2d 6363 2d2d 6464 2d2d aa--bb--cc--dd--
00000010  891d e000 d803 5f61 581b 0d61 4fd8 d4ac ..`.X._aX..aOXT,
00000020  0700 0100 0000 0000 f8fa 2200 0000 0000 ........xz".....
00000030  e538 f877 0000 2300 68c7 2300 0000 0000 e8xw..#.hG#.....
00000040  10ca dc77 40b8 dc77 0000 0000 8a01 0000 .J¥w@8¥w........
00000050  3cfb 2200 c876 f877 aafb 2200 3502 0000 <{".Hvxw*{".5...
00000060  0000 d877 3cc1 dc77 10ca dc77 a8fb 2200 ..Xw<A¥w.J¥w({".
00000070  0100 0000 0000 0000 c800 d877 0cfb 2200 ........H.Xw.{".
00000080  10ca dc77 58fb 2200 0b00 0000 7039 f877 .J¥wX{".....p9xw
00000090  0000 2300 9808 2300 0b00 0000 0000 0000 ..#...#.........
000000a0  30fb 2200 0002 0000 0000 2300 10cb 2300 0{".......#..K#.
000000b0  7801 2300 9e00 0000 50b6 fc77 16b5 fc77 x.#.....P6|w.5|w
000000c0  2db5 fc77 00ec fd7f f8eb fd7f 0000 0000 -5|w.l}.xk}.....
000000d0  0000 2300 b8ca 2300 7801 2300 a900 0000 ..#.8J#.x.#.)...
000000e0  50b6 fc77 16b5 fc77 2db5 fc77 7801 2300 P6|w.5|w-5|wx.#.
000000f0  7801 2300 442b 4000 0000 0000 4006 2300 x.#.D+@.....@.#.

  Addr     0 1  2 3  4 5  6 7  8 9  A B  C D  E F 0 2 4 6 8 A C E
--------  ---- ---- ---- ---- ---- ---- ---- ---- ----------------
00000100  e800 0000 0099 0d61 0300 0000 f8eb fd7f h......a....xk}.
00000110  0000 0000 7801 2300 7801 2300 b57c f877 ....x.#.x.#.5|xw
00000120  e8fb 2200 4006 2300 e400 0000 0000 2300 h{".@.#.d.....#.
00000130  0300 0000 0000 0000 0000 0000 0000 0000 ................
                          ～中略～
000003a0  1030 4000 46f0 0ddf 68fe 2200 f6b4 0861 .0@.Fp._h~".v4.a
000003b0  f803 040a 60fe 2200 68fe 2200 5014 4000 x...`~".h~".P.@.
000003c0  2000 040a 442b 4000 88fe 2200 9814 4000  ...D+@..~"...@.
000003d0  e003 040a 0050 4000 0000 0000 0000 0000 `....P@.........
000003e0  402b 4000 a02a 4000 98fe 2200 5f10 4000 @+@. *@..~"._.@.
000003f0  0040 4000 0050 4000 b8fe 2200 915d 0061 .@@..P@.8~"..].a
```

〔リスト17〕リスト16の実行結果データダンプリスト

```
  Addr     0 1  2 3  4 5  6 7  8 9  A B  C D  E F 0 2 4 6 8 A C E
--------  ---- ---- ---- ---- ---- ---- ---- ---- ----------------
00000000  6161 2d2d 6262 2d2d 6363 2d2d 6464 2d2d aa--bb--cc--dd--
00000010  891d e000 d803 5f61 581b 0d61 4fd8 d4ac ..`.X._aX..aOXT,
00000020  0700 0100 0000 0000 f8fa 2200 0000 0000 ........xz".....
00000030  e538 f877 0000 2300 68c7 2300 0000 0000 e8xw..#.hG#.....
00000040  10ca dc77 40b8 dc77 0000 0000 8a01 0000 .J¥w@8¥w........
00000050  3cfb 2200 c876 f877 aafb 2200 3502 0000 <{".Hvxw*{".5...
00000060  0000 d877 3cc1 dc77 10ca dc77 a8fb 2200 ..Xw<A¥w.J¥w({".
00000070  0100 0000 0000 0000 c800 d877 0cfb 2200 ........H.Xw.{".
00000080  10ca dc77 58fb 2200 0b00 0000 7039 f877 .J¥wX{".....p9xw
00000090  0000 2300 9808 2300 0b00 0000 0000 0000 ..#...#.........
000000a0  30fb 2200 0002 0000 0000 2300 10cb 2300 0{".......#..K#.
000000b0  7801 2300 9e00 0000 50b6 fc77 16b5 fc77 x.#.....P6|w.5|w
000000c0  2db5 fc77 00ec fd7f f8eb fd7f 0000 0000 -5|w.l}.xk}.....
000000d0  0000 2300 b8ca 2300 7801 2300 a900 0000 ..#.8J#.x.#.)...
000000e0  50b6 fc77 16b5 fc77 2db5 fc77 7801 2300 P6|w.5|w-5|wx.#.
000000f0  7801 2300 342c 4000 0000 0000 4006 2300 x.#.4,@.....@.#.

  Addr     0 1  2 3  4 5  6 7  8 9  A B  C D  E F 0 2 4 6 8 A C E
--------  ---- ---- ---- ---- ---- ---- ---- ---- ----------------
00000100  e800 0000 0099 0d61 0300 0000 f8eb fd7f h......a....xk}.
00000110  0000 0000 7801 2300 7801 2300 b57c f877 ....x.#.x.#.5|xw
00000120  e8fb 2200 4006 2300 e400 0000 0000 2300 h{".@.#.d.....#.
00000130  0300 0000 0000 0000 0000 0000 0000 0000 ................
                          ～中略～
000003a0  1030 4000 46f0 0ddf 68fe 2200 f6b4 0861 .0@.Fp._h~".v4.a
000003b0  f803 040a 60fe 2200 68fe 2200 2015 4000 x...`~".h~". .@.
000003c0  2000 040a 342c 4000 88fe 2200 6815 4000  ...4,@..~".h.@.
000003d0  e003 040a 0050 4000 0000 0000 0000 0000 `....P@.........
000003e0  302c 4000 902b 4000 98fe 2200 5f10 4000 0,@..+@..~"._.@.
000003f0  0040 4000 0050 4000 b8fe 2200 915d 0061 .@@..P@.8~"..].a
```

〔リスト16〕vsnprintfの例(test136.c)

```
#include <stdarg.h>
/*
 *vsscanfについて
 */
#include <stdio.h>
void func(int num, ...);
main()
{
    func(4,"aa","bb","cc","dd");
}
void func(int num, ...)
{
    FILE *out1;
    FILE *out2;
    va_list ap;
    va_list dup_ap;
    va_list dup_ap1;
    char *str;
    char buf[1024];
    char *data;
    int  ix;
    va_start(ap, num);
    __va_copy(dup_ap,ap);
    vsnprintf(buf,10,"%s:%s:%s:%s:%s",ap);
    out1 =    fopen("test1.dat","w");
    fwrite(buf,1,1024,out1);
    fclose(out1);
    vsnprintf(buf,20,"%s--%s--%s--%s--%s--%s",dup_ap);
    out2 =    fopen("test2.dat","w");
    fwrite(buf,1,1024,out2);
    fclose(out2);
    vsscanf(buf,"%s--%s--%s--%s--%s",dup_ap1);
    va_start(dup_ap1, num);
    data =    va_arg(dup_ap1,char *);
    printf("%s¥n",data);
    data =    va_arg(dup_ap1,char *);
    printf("%s¥n",data);
    data =    va_arg(dup_ap1,char *);
    printf("%s¥n",data);
    data =    va_arg(dup_ap1,char *);
    printf("%s¥n",data);
    va_end(ap);
    va_end(dup_ap);
    va_end(dup_ap1);
}
```

〔リスト18〕strtofの例(test137.c)

```
#include <stdarg.h>
/*
 *strtofについて
 */
#include <stdio.h>
#include <stdlib.h>
void func(int num, ...);
main()
{
    float            wk1;
    long     double   wk2;
    const    char     test1[]   =     "0.1236548";
    const    char     test2[]   =
"98612665464.545236548";
    wk1          =    strtof(test1,NULL);
//  wk2          =    strtold(test2,NULL);
    printf("%f¥n",wk1);
//  printf("%Lf¥n",wk2);
}
```

● stdint.hで定義された型を使う方式について

これはstdint.hで新たに定義された型を標準ライブラリで使用する際に，マッチングをとるための書式指定マクロの定義です．inttypes.hに定義されているはずですがGCCではstdint.hに対応していないので，書式指定マクロもまだ定義されていません．

● stdbool.hについて

これはGCC3.2の環境で動作しました．stdbool.hにはbool，true，falseが定義されていて，プログラム中で使うことができます(**リスト21**)．

実行結果は以下のとおりです．

```
$ ./test140
```

〔リスト 19〕strtoll, strtoull の例(test138.c)

```
#include <stdarg.h>
/*
 *strtoll について
 */
#include <stdio.h>
#include <stdlib.h>
main()
{
    long long     int                 wk1;
    long long     unsigned  int       wk2;
    const         char      test1[]   =    "-965485412";
    const         char      test2[]   =    "98612464";
    wk1           =    strtoll(test1,NULL,10);
    wk2           =    strtoull(test2,NULL,10);
    printf("%ld¥n",wk1);
    printf("%ld¥n",wk2);
}
```

〔リスト 20〕atoll の例(test139.c)

```
#include <stdarg.h>
/*
 *strtoll について
 */
#include <stdio.h>
#include <stdlib.h>
main()
{
//  lldiv_t       res;
    long long     int            wk1;
    long long     int            wk2;
    const         char           test1[]   =    "98612464";
//  wk1   =    atoll(test1);
    wk1   =    -653587458;
    wk2   =    llabs(wk1);
    printf("%ld¥n",wk2);
//  res   =    lldiv(wk2, wk1);
//  printf("%d¥n",res.quot);
//  printf("%d¥n",res.rem);
}
```

〔リスト 21〕stdbool.h の例(test140.c)

```
/*
 *stdbool.h について
 */
#include <stdio.h>
#include <stdbool.h>
main()
{
        bool   test;
        test =    true;
        if   (test     )
        {
            printf("test は真です¥n");
        }
}
```

〔図 1〕Status of C99 features in GCC

Status of C99 features in GCC - GNU Project - Free Software Foundation

http://gcc.gnu.org/c99status.html

Status of C99 features in GCC

This table is based on the list in the foreword to the printed standard ISO/IEC 9899:1999 (E) as amended by ISO/IEC 9899:1999/Cor 1:2001 (E)

Where "Library Issue" is listed in conjunction with some other status, this means that some compiler support is needed for the library support, or desirable in conjunction with it. Note that the headers required of conforming freestanding implementations clause 4 paragraph 6, do not count as library issues

This page describes the C99 support in mainline GCC CVS, not in any particular release. Information is also available on C99 support in GCC 3.3, C99 support in GCC 3.1 and 3.2 and on C99 support in GCC 3.0 but not on the much more limited support in GCC 2.95

See below the table for further notes on some issues

Feature	Library Issue	Done	Broken	Missing
restricted character set support via digraphs and `<iso646.h>` (originally specified in AMD1)		Done		
wide character library support in `<wchar.h>` and `<wctype.h>` (originally specified in AMD1)	Library Issue			Missing
more precise aliasing rules via effective type		Done		
restricted pointers		Done		
variable-length arrays			Broken	
flexible array members		Done		
`static` and type qualifiers in parameter array declarators		Done		
complex (and imaginary) support in `<complex.h>`			Broken	
type-generic math macros in `<tgmath.h>`	Library Issue	Done		
the `long long int` type and library functions		Done		

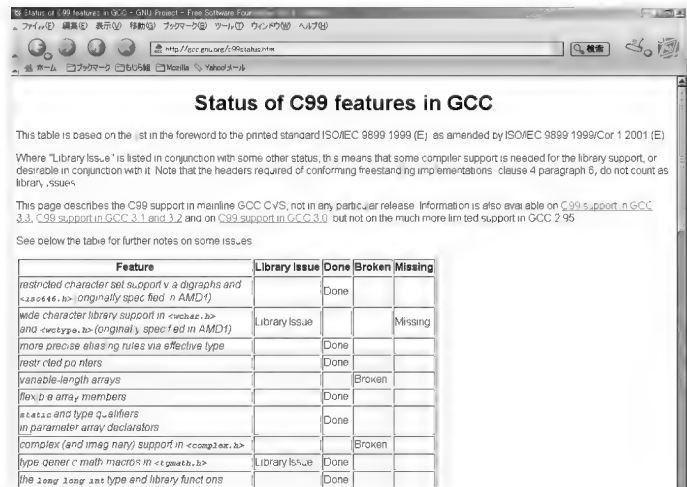
test は真です

- stdint.h について

前述しましたが stdint.h は定義されていません．現在 int 型の大きさは各環境によって 8 ビットだったり，32 ビットだったりしますが，これを共通に扱うようにする目論見です．しかし，これは大きな変更なので受け入れられるかどうかわかりません．

たとえば，int32_t という型をこの環境にもっていっても 32 ビットと扱えば，int 型の大きさに依存するより，各環境に移植しやすくなります．

- tgmath.h について

これも定義されていません．たとえば数学関数では，double を返す sin()，float を返す sin()，long double を返す sin() がありますが，これをマクロを使って見かけ上一つの関数にするためのものです．

- C99 規格対応に関する GNU C のバージョンごとの進捗

いろいろ問題はあるようですが，C99 規格の重要な部分は GCC3.2 で対応できているようです．これに関する公式情報は以下のサイトにあるので，参照してください(**図 1**)．

http://gcc.gnu.org/c99status.html

このページは GCC2.95 の対応についてまとめてありますが，このページから GCC3.0，GCC3.1，GCC3.3 の対応をまとめたページへリンクが張ってあるので参考にしてみてください．

＊　＊

C99 規格の説明と GCC の対応については，これで終わります．

きし・てつお

Interface3 月号増刊　好評発売中

組み込みエンジニアのための
Embedded UNIX Vol.2

A4 変型判　192 ページ
定価 1,490 円(税込)

- 第 1 特集 作りながら学ぶ 組み込み Linux システム設計
- 第 2 特集 UNIX として設計された RTOS —— LynxOS

CQ出版社　〒170-8461 東京都豊島区巣鴨 1-14-2　販売部 TEL.03-5395-2141　振替 00100-7-10665

Webサーバ機能をもつEthernet-シリアルコンバータ「XPort」活用技法　前編

シリアル機器をEthernetに接続する

川口幸裕

ユビキタス時代の到来とシリアル通信機器

組み込み機器分野におけるEthernetの普及がめざましい．さまざまな機器にネットワーク機能が盛り込まれ，パソコン(PC)との通信だけではなく，機器間での通信，1対多，多対1，多対多形態での接続があたりまえとなっている．また，Ethernetコントローラを内蔵したCPUが，組み込み機器分野でも数多く使われている．

しかし，安価で単純な機能しかもたない組み込み機器では，まだシリアル通信が主流である．Ethernetコントローラをもたず外部との通信手段としてシリアルを標準装備したCPUは，安価で扱いやすい．また歴史が古いため，蓄積された多くのリソースが存在し，おいそれとそれらを放棄するわけにいかない現状もある．しかし，Ethernetの普及を指をくわえて見ているわけにもいかない．

組み込み用シリアル⇔Ethernetコンバータ「XPort」

この現状に，一石を投じる製品が発表された．2003年2月24日，米国ラントロニクス社は，**RJ-45コネクタ大の大きさ**にCPU，メモリ，ファームウェア，RJ-45コネクタを装備したメディアコンバータ「ナノサイズデバイスサーバXPort」を発表したのである．

このXPortはシリアル⇔Ethernet変換を基本機能とし，Webサーバをも内蔵している．詳細スペックは**表1**を参照いただきたい．

XPortには，シリアル機器をEthernetに接続するために必要なハードウェア/ソフトウェアがすべて搭載されている．このことは，既存もしくは新たに開発されるシリアル機器においてデータ入出力部位のアーキテクチャ変更が不要であることを意味し，またシリアル機器側のCPUをはじめとした各種資源への負荷が非常に小さいことも特徴としてあげられる．内包されたソフトウェアには組み込み業界で定評のある旧US Software社(2000年12月にラントロニクス社が買収)の製品が採用されている(**図1**)．

また，必要なソフトウェアをすべて搭載することで，元来ユーザーが行うべきソフトウェア類の開発，移植作業をかぎりなくゼロに近づけた．基本的には，既存のシリアル機器側のソフトウェアに関する変更は必要ない．

いままでも機能的にXPortに類する機器は数多く存在する．ボックス製品，小さなボード形状のもの(ラントロニクス社ではボックス製品，ボード製品も取り扱っている)など．しかしXPortは設計段階で，組み込み機器への搭載を前提に開発されている．

〔表1〕XPortの詳細スペック

CPU	80186ベースCPU 48MHz 12.5MIPS
メモリ	256Kバイトノーウェイト SRAM
	512Kバイトフラッシュメモリ
I/O	300bps～230kbpsシリアル(CMOS 3.3V)
Ethernet	10/100Base-T Ethernet
LED	10/100Base-T状態，Full/Half状態
消費電力	DC 3.3V 210mA
動作温度	－40℃～＋85℃

XPort評価キットの概要

製品発表にともないXPortの評価キット(**写真1**)を使用する機会を得た．とくに難しい技術も必要なく，シリアル機器を

〔図1〕XPortのソフトウェア

＊Rijndael：秘密鍵方式の暗号システム

〔写真1〕XPort評価キット

Ethernetに簡単に接続できる評価ボードとケーブル類，ソフトウェアを同梱したキットである．

今回はこのXPort評価ボードを使って，シリアル機器を実際にEthernetに接続し，PCからのコントロールを行ってみた．ここではその工程を誌上シミュレーションという形で記述してみる．

● XPort評価キットの内容

評価キットには以下のものが同梱されている．

▶ XPort評価ボード
▶電源アダプタ
▶電源アダプタ用コネクタ4種(各国対応)
▶ Ethernetケーブル(ストレート)
▶ RS-232-Cケーブル(ストレート)
▶ D-Sub25ピン⇔D-Sub9ピン変換アダプタ
▶ CD-ROM

● 必須機器

▶Windows2000/XP/Me/98が動作する，Ethernet接続可能なPC

後述するDevice Installerでのネットワークアダプタの認識を考えた場合，2000/XPが望ましい．ノートPC，デスクトップPCなど，形状はとくに問わない．

〔表2〕XPortのRS-232-C部のピンアサイン

D-Sub9ピン	説　明
1	DTR
2	TXD(Data Out)
3	RXD(Data In)
4	CTS/DCD
5	Ground
6	Not Connected
7	CTS/DCD
8	RTS
9	Not Used

● その他必要な機器

▶ Ethernetクロスケーブル

実際にXPort評価ボードを使ってシリアル機器を接続する際には，Ethernetハブを介さず，テスト用PCとXPort評価キットを直結させたほうが都合が良い場合もあると思われる．そこで，Ethernetクロスケーブルを用意しておくとよいだろう．

▶ RS-232-Cクロスケーブル

表2のとおり，XPortのRS-232-C部のピン割り当てはクロス接続となっている．評価対象のシリアル機器とPCとの間にクロスケーブルを使用していた場合は，ストレートケーブル(当キットに付属)による接続，シリアル機器とPCとの間にストレートケーブルを使用していた場合はクロスケーブル(もしくはクロスコネクタ/アダプタ)が必要になる．

▶ D-Sub25ピン⇔D-Sub9ピン変換アダプタをもう1個
▶ D-Sub9ピンのオス⇔オス，ジェンダーチェンジャを2個
▶ D-Sub9ピンのメス⇔メス，ジェンダーチェンジャを2個

RS-232-Cシリアル機器をXPort評価ボードに接続する際，これらも用意しておいたほうが望ましい．

● CD-ROMの内容

現在添付されているCD-ROMには，CD-XPT-02 Rev.Aと銘打たれている．本リビジョンには下記の内容が収められている．

▶ XPortドキュメント

● XPort Fact Sheet

CD-ROMに収められた機能カタログはプレリミナリバージョンである．最新のものは，ラントロニクス社のホームページ(http://www.lantronix.com/)から入手できる．

● XPort Data Sheet

これも上記同様，プレリミナリデータシートである．本稿執筆段階ではラントロニクス社のホームページからは入手できないが，本号の発売時期には入手可能なはずである．

● XPort評価キットマニュアル

本稿で使用するXPort評価ボードのマニュアル

▶ Device Installer

評価ボード上のXPortの各種設定を行うためのWindows用GUIツール．詳細は後述する．

▶ ComPort Redirector

Windows用，仮想COMポートドライバ(詳細は後述)．

● PCの準備とXPort評価ボードの設定

PCには，Device Installerをインストールしておく．Device InstallerをインストールしたPCとXPort評価ボードは，Ethernetで接続し通信可能な状態にする．

具体的には，

1) XPort評価ボードへEthernetケーブルを接続する(ストレートケーブルを使用しハブ経由，もしくはクロスケーブルを使ってPCと直結)．
2) XPort評価ボードの電源を投入する．
3) Device Installerを起動する．

4) Device Installerにて Searchを行う．ネットワーク上のXPort評価ボードがリストアップされる．
5) XPort評価ボードへのIPアドレス設定を行う．XPortへのIPアドレスの設定方法は3種ある．Device Installerより，Assign IPボタンを押下しIPアドレスを入力する．Telnetを使用し，コマンドラインにて設定を行う．もしくは，Device InstallerのWebボタンを使用し，Webブラウザを起動して設定を行う．XPortはWebサーバ機能をもっているが，デフォルトのWebページとして，Web Managerと呼ばれる設定ツールがインストールされている．これは，Webブラウザを通して，IPアドレス，RS-232-Cの各種設定を行うことができる．Assign IPもしくは，Webによる設定を行うのが無難であろう．
6) リセット＆リブート．5)の手順で，Telnet以外を使えば自動でリセット＆リブートが行われる．Telnetで変更した場合，ボード上のリセットボタンを押下すればよい．
7) 再度，Searchを行い，IPアドレスが変更されていることを確認する．

準備は以上である．

● XPort Installer

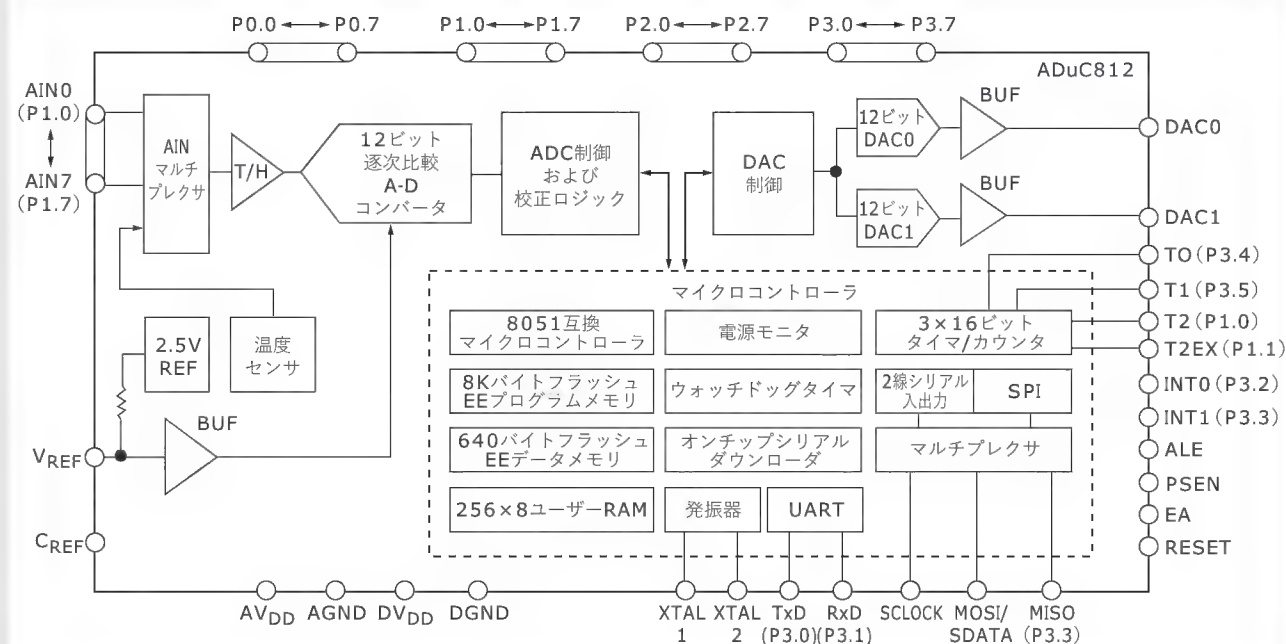
Lantronix XPort Installerは，評価キットに標準添付されているWindows上のGUIソフトウェアである．現時点での名称はDevice Installerだが，正式版がリリースされたときには，XPort Installerに変更される予定である．おもな機能は次のとおりである[注1]．

注1：Windows 95/98/Me上で，ごくまれに，XPort Installerにて認識できないネットワークインターフェースが存在する．この場合，別のネットワークインターフェースを用意する必要がある．

マイクロコンバータ

今回，仮想シリアル機器として使用した，アナログ・デバイセズ製マイクロコンバータ評価ボードEVAL-ADuC812QS（**図A**）には，ADuC812マイクロコンバータ（マルチチャンネル，フラッシュMCU埋め込み12ビットA-Dコンバータ，D-Aコンバータ内蔵）が搭載されている．マイクロコンバータは高精度A-D，D-Aコンバータと8ビットプログラマブルMCU（インテル8052互換），DMAコントローラ，SRAM，フラッシュメモリ，シリアルコントローラ（UARTなど）を1チップ化したデータアクイジション（取得）システムである．

1チップ化したことで，D-A，A-Dコンバータ，MCU間での配線ノイズが大幅に軽減され，A-Dコンバータパワー投入時のノイズキャリブレーションによって，より精度の高いデータを得ることができる．また元来，非常に煩雑であるD-A，A-Dコンバータの制御が，非常に簡単なプログラムで実現できる．マイクロコンバータはインテリジェントセンサ，バッテリ駆動システム，データ収集システム，計測システムおよび通信システムなど，多様な分野で使用されている．

〔図A〕マイクロコンバータ評価ボードEVAL-ADuC812QSの機能ブロック図

- 検索(ネットワーク上のXPortを検索，リストアップする)
- 各XPortへのIPアドレスの設定
- ファームウェアのアップデート/リカバリ(COMポート経由)
- Webページの変換とアップロード(Ethernet経由)
- ファームウェアのアップデート/リカバリ(Ethernet経由)
- Telnetクライアントの呼び出し
- Webブラウザの呼び出し
- Pingクライアントの呼び出し

2台のPC間の通信

まず最初に，XPort InstallerをインストールしたPC(以下，テストPC)側のTelnet端末と，Windows標準ターミナルソフトであるHyperTermもしくはその他の端末ソフトウェア間での文字通信を行ってみよう．

テストPCともう1台，RS-232-CポートをもったPC(以下，端末PC)を用意する(テストPCと端末PCはネットワーク接続可能で，RS-232-Cポートをもった1台のPCで代用してもかまわない．

テストPCとXPort評価ボードは「PCの準備とXPort評価ボードの設定」のところで述べたとおり，Ethernet接続されているものとする．

XPort評価ボードのRS-232-Cポートと端末PCをXPort評価キットに付属しているストレートケーブルで接続する．

端末PC側で，HyperTerminal(コンソール端末ソフトウェア)を起動する．使用するCOMポートは，ケーブルを接続したポート番号，ボーレートは9600，データ8ビット，パリティなし，ストップ1ビット，フロー制御なし，で設定する．

続いて，テストPC側でDevice Installerを起動し，Telnetボタンを押下する．TCP/IPのポート番号を入力するダイアログが現れる．9999がセットされているが(9999はTelnetでXPortの設定を行うためのポート番号)，ここに10001を入力する．10001はデフォルトで設定されている，Telnet端末とRS-232-Cポートを直結してあるポートで，前述のWeb ManagerのLocal Portもしくは，Telnet端末(ポート番号9999)を使用して，変更できる．ここではとくに変更せず，10001をそのまま使用する．

テストPC側にTelnet端末ウィンドウが表示される．文字を入力してみると，端末用PCのHyperTerminalの画面に入力した文字が表示されるはずである．日本語も問題なく入力，表示される．

テストPC側の入出力はEthernet，端末PC側の入出力はRS-232-Cが使用され，相互通信が何ら問題なく，またXPortの設定をとくに変更する必要もなく(当然，専用プログラムを記述する必要なく)，接続されたことが確認できる．

この場合の端末PCはそのまま，シリアル機器に置き換えて考えていただけるとわかりやすいと思う．シリアル機器側の入出力には何ら手を加えず，Ethernetを通して，テストPCからコントロールしているのである．

〔リスト1〕評価ボード側にダウンロードしたプログラム

```
/*
    a812uart.c

    Analog Devices 'MicroConverter' ADuC812 Simple Sample Program
                   D.Hagiya, Y.Kawaguchi, Tachibana Tectron, 2003.3.12
    ADuC812-QS A/D Spec       Rate : 20K SPS, Resolution : 12 bit
*/

#include <stdio.h>
#include <ADuC812.h>

int v;

void ADC () interrupt 6
{
    int u, l;

    u = ADCDATAH & 0x0F;
    l = ADCDATAL & 0xFF;
    v = (u << 8) + l;
}

void main(void)
{
    SCON = 0x52;        /* set baudrate : 9600bps           */
    TMOD = 0x20;
    TH1  = 0xFD;
    TR1  = 1;
    ADCCON1 = 0x70;     /* A/D Converter Mode : Normal, 20K */
    ADCCON2 = 0x00;     /* A/D input channel                */
    EADC=1;             /* interrupt Enable (A/D converter) */
    EA=1;               /* interrupt Enable (MCU)           */

    CCONV = 1;          /* A/D Convert Start                */

    while(1)
    {
        printf("%d¥n", v);
    }

    CCONV  = 0;         /* A/D converter stop */
}
```

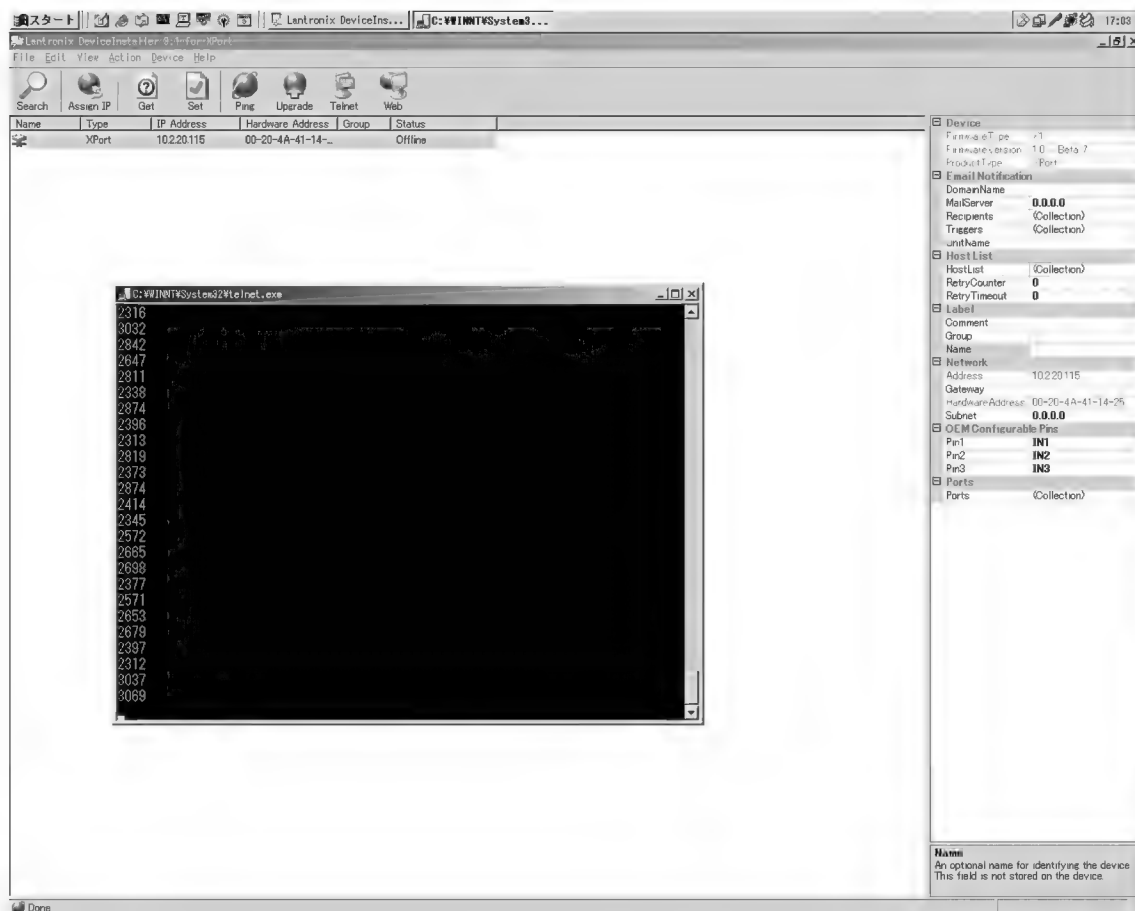
マイクロコンバータとの接続

それではいよいよ，実際のシリアル機器をEthernet経由でコントロールしてみよう．今回シリアル機器として用意したのは，アナログ・デバイセズのマイクロコンバータ評価ボードEVAL-ADuC812QSである．マイクロコンバータについては，**コラム1**を参照いただきたい．

マイクロコンバータ評価ボード側には非常に簡単なプログラムをダウンロードしておいた(リスト1)．

このプログラムはマイクロコンバータ評価ボード上にあるアナログ信号入力をマイクロコンバータ内のA-Dコンバータで変換し，その変換結果を10進数でRS-232-Cに出力しているだけである(誌面の都合上プログラムはあえて短くしたが，筆者が実際のデモンストレーションで使用しているプログラムは，出力データフォーマットを変えたり，コンバートトリガ，回数指定，メモリ参照などができるようになっている)．

前もってこのプログラムの出力結果を端末PCで確認しておくのもいいだろう．マイクロコンバータ評価ボードのRS-232-CポートはXPort評価ボードのRS-232-Cポートと同様，クロス割り

〔図 2〕テスト PC の Telnet 端末ウィンドウの表示例

当てになっているため，RS-232-C ストレートケーブルを使用して接続する．マイクロコンバータ評価ボードの電源を入れると，4 桁の 10 進数が連続して HyperTerminal に表示される．

マイクロコンバータ評価ボードと XPort 評価ボードの RS-232-C ポートを接続する．ここでは，RS-232-C クロスケーブルを使用する．

接続が終わったらテスト PC で先ほどと同様に，Telnet 端末ウィンドウを起動しておく．

マイクロコンバータ評価ボードの電源を入れてみると，端末 PC の HyperTerminal に出力されたものと同じものがテスト PC の Telnet 端末ウィンドウに表示される（図 2）．

実際のデモでは，テスト PC から，データコンバートの開始，表示データのフォーマット指定などのコマンドを送ることが可能で，シリアル機器を Ethernet 経由で完全にコントロールできることを実証している．

このことからマイクロコンバータを，ある CPU の制御下で動作しているシリアル機器と考えていただきたい．その CPU には当然 Ethernet をコントロールするだけの処理能力はなく，そのためのメモリなどは搭載されていない．しかし，XPort を接続するだけで，そのシリアル機器は Ethernet 対応機器として動作させられる．XPort は非常に小型なため，外付け機器としてではなく，組み込み用機器として，対象となるシリアル機器に搭載できるのである．

仮想 COM ポートソフトウェア「ComPort Redirector」と，加速度センサの接続

● 仮想 COM ポートソフトウェア「ComPort Redirector」とは？

いままでのテスト用 PC 側での Telnet 端末ウィンドウを使用した方式では，あくまでも文字データ転送を主としていた．たとえばバイナリデータの転送を前提としたグラフィック表示アプリケーションなどは，先ほどの Telnet 端末用ポートを使用した Ethernet 用アプリケーションに作り直さなければならない．

ラントロニクス社はシリアル機器側の Ethernet 接続ソリューションとしての XPort のみではなく，WindowsPC 側の XPort 対応ソリューションとして ComPort Redirector というソフトウェアを用意している．これは XPort 評価キットの CD-ROM に内包されている．

ComPort Redirector は仮想 COM ポートを Windows 上に生成し，Ethernet を物理層とした入出力ドライバである．この

ComPort Redirectorを利用すれば，XPortによってEthernet接続されたシリアル機器に対するWindows側のコントロールアプリケーションをネットワークアプリケーションに書き直す必要はない．

それでは実際にComPort Redirectorを使って，シリアル機器をつなげてみる．

● 加速度センサの接続

シリアル機器として用意したのは，アナログ・デバイセズ社の加速度センサ評価ボードADXL202EB-232-Aである．加速度センサについては**コラム2**を参照いただきたい．

この加速度センサ評価ボードは通常，WindowsPCのRS-232-Cに接続する．また，この評価ボードに付属しているツールは，加速度センサから送られてくるデータを視覚的に見ることができるもので，通常のCOMポートからのバイナリデータをグラフィック表示する．

テストPCにComPort Rediretorをインストールする．ComPort Redirector Configurationプログラムが同じく起動可能となるので，このプログラムを立ち上げ，仮想COMポートの番号，XPortのIPアドレス，Telnet用ポート番号を設定する．設定を変更した後，テストPCは再起動する必要がある．ここではCOM2を使用することにする．

続いて，XPortのRS-232-C側の通信速度を変更する．この加速度センサは38400bps固定なので，Web Managerを使って，XPortのRS-232-C側を38400に変更する．Telnetポート9999を使ってもかまわない．

次に，テストPCに加速度センサ用のデータ表示ツールであるX-Analyzeをインストールする．これは加速度センサ評価ボードに添付されているCD-ROMに同梱されている．X-Analyzeを起動し，接続を確立するための設定を行う．先ほど，ComPort Redirector Configurationにより指定したCOMポートを指定する．X-AnalyzeのコネクションツールはWindows上のCOMポートをすべて検索するので，COM2が自動で選択できるはずである．このとき，ComPort RedirectorがConfigurationで指定したIPアドレスとポート番号を接続ダイアログで表示する（ComPort Redirectorは設定されたポート番号へのアクセス時に自動でネットワーク接続を行う）．

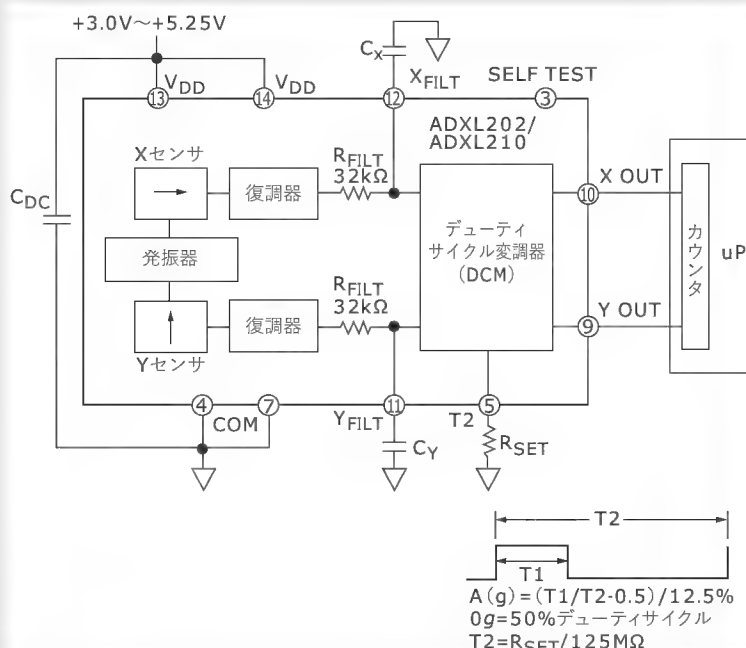
X-AnalyzeへのCOMポートの設定が終わったら，同じく，Configure a Connectionを使用し，加速度センサにセンシングコマンドを送るタイミングを変更する．デフォルトではAs fast as possibleになっているので，User-selected rateに変更し，Update rateには65Hz以下を指定する．

ここは非常に重要である．As fast as possibleではRS-232-Cに加速度センサが直結されているという前提で，センシングタイミングを作るのだが，XPortおよびComPort Redirectorを使用する場合，ネットワーク通信のオーバヘッドがかかってしまう．もちろんXPortの送受信タイミングを変更することで，このオーバヘッドは小さくできるのだが，現在のWeb Managerでは，送信タイミングは最小で12msとなっている．これだと最大のUpdate Rateは83Hzとなる．しかも，テストPCからのセンシングコマンドの送信も必要なので，65Hzぐらいが安定した入

加速度センサ

今回，仮想シリアル機器として使用した，アナログ・デバイセズ製加速度センサ評価ボードADXL202EB-232-Aには，iMESディジタル出力付き加速度センサADXL202（**図B**）が搭載されている．加速度センサADXL202は低パワーの2軸加速度センサで，振動などの動的加速度も重力などの静的加速度も，ともに計測可能である．

また，このセンサの出力はマイクロプロセッサのカウンタで直接測定でき，A-Dコンバータやロジック回路を必要としない．測定範囲は−2G〜+2Gと幅広く，また，出力周期も0.5ms〜10msの範囲に調整できる．iMEMS加速度センサは，2軸傾斜センサ，コンピュータ周辺機器，慣性ナビゲーション，地震モニタ，車両安全システムおよびバッテリ駆動動作センサなど，幅広いアプリケーションで使用されている．

〔図B〕ADXL202

出力を得ることができる最大値となる．今後SDKがリリースされ，XPort内のプログラムを変更することで，この送信タイミングを変更することは可能になると思われる．

すなわち，WindowsPC側のアプリケーションでタイムアウト処理を行っている場合，そのタイムアウトの調整が必要になる．これはComPort Redirectorを使う上で，唯一気をつけなければいけない点である．

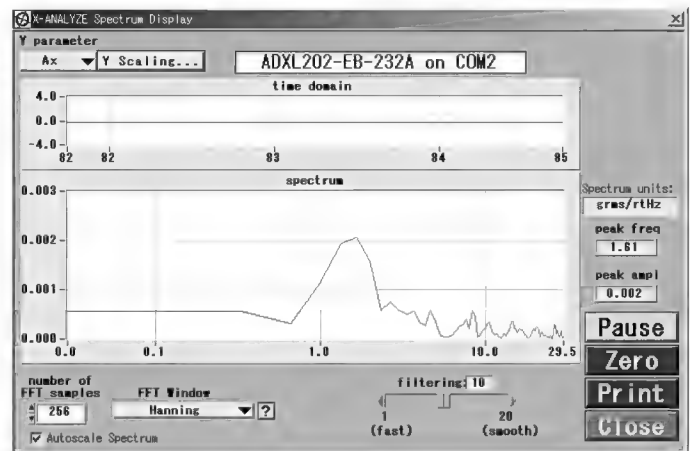
それでは実際に接続する．X-Analyzeには3種類のデータ表示ツールが用意されているが，今回は線形表示ツールを選んだ(**図3**)．

加速度センサ評価ボードを動かすと，**図3**の波形が変化する．加速度センサのようなCPUやメモリをもたないものでも，シリアル出力さえできれば，ネットワークに接続できる．

もちろん，X-Analyzeそのものには何も変更を加えていない．唯一の変更は，COMポートの番号だけである．Windows側のアプリケーションも，実際は何も変更する必要はない．

〔図3〕X-Analyzeの線形表示ツール

おわりに

XPortとComPort Redirectorの組み合わせにより，既存のシリアル機器を非常に簡単にEthernet対応にできる．また，シリアル機器をEthernet対応機器とすることで，多くの利点が生まれる．RS-232-CケーブルとEthernetケーブルの違い，ケーブル接続の取り回し，切り替え機とEthernetハブの違いも大きいであろう．複数のシリアル機器を1台のPCで集中管理することが可能となる．また，1台のシリアル機器を複数のPCで共有することもできる．計測機器，制御機器，小型機器などさまざまな分野でいまだにシリアル機器は数多く使われている．実際，多くのシリアル⇔Ethernetコンバータや，それに類するソリューションがある．しかしXPortには，超小型であること，プログラマブルであること，RedirectorによるWindows側のアプリケーションの変更/作成不要という大きなアドバンテージがある．このXPort評価キットを使って，手元のシリアル機器をEthernetにつないでみてほしい．きっと新しいアイデアが浮かんでくるのではないだろうか．

*　　　*

今回は，評価キットを使って既存のシリアル機器とPCとの接続を行ってみた．次回は，Javaアプレットを使ったWebページでのシリアル機器のコントロール，SDKの解説を予定している．

■**「XPort」に関する問い合わせ先**
橘テクトロン(株)
・営業関係問い合わせ先：sales.lantronix@tachitec.co.jp
・技術的ご質問：support.lantronix@tachitec.co.jp

かわぐち・ゆきひろ　橘テクトロン(株)

Interface BackNumber

2002年
5月号 CD-ROM付き オブジェクト指向の実装技法入門
6月号 別冊付録付き これでわかる！マイクロプロセッサのしくみ
7月号 別冊付録付き Linux徹底詳解 ― ブート&ルートファイルシステム
8月号 CD-ROM付き 組み込み分野へのBSDの適用
9月号 別冊付録付き 基礎からの計算科学・工学 ― シミュレーション
10月号 データベース活用技術の徹底研究
11月号 CD-ROM付き 徹底解説！ ARMプロセッサ
12月号 多国語文字コード処理&国際化の基礎と実際

2003年
1月号 別冊付録付き 作りながら学ぶコンピュータシステム技術
2月号 CD-ROM付き ワイヤレスネットワーク技術入門
3月号 ICカード技術の基礎と応用
4月号 別冊付録付き 解説！ USB徹底活用技法
5月号 CD-ROM付き うまくいく！組み込み機器の開発手法

CQ出版社　〒170-8461 東京都豊島区巣鴨1-14-2　販売部 ☎(03)5395-2141　振替 00100-7-10665

第19回

XMLとJavaScriptを解釈・実行するランタイムエンジン ―Konfabulator

水野貴明

今回紹介するKonfabulatorは，Mac OS Xで動作する「Widget」と呼ばれる小さなアプリケーションを実行するためのランタイムエンジンである．つまり，Konfabulator自体に，ユーザーに訴えかける何らかの機能があるわけではなく，たとえばFlashの再生にFlashプレーヤが必要であるように，KonfabulatorはWidgetというプログラムを実行するために，利用される実行環境なのである．

Konfabulatorは，20003年の2月に公開されて以来，非常に注目されており，今後Mac OS Xにおける定番ソフトウェアとなる可能性を秘めている．そこで今回は，Konfabulatorのしくみと Widgetのプログラミング，そしてKonfabulatorが注目される理由について，見ていくことにする．

インストールとWidgetの実行

Konfabulatorは，Mac OS Xのディスクイメージ(dmg)形式のファイルとして配布されている．ディスクイメージをマウントすると，中にはKonfabulatorのファイル本体だけが入っているので，それをどこか好きな場所にコピーして起動すれば，インストールは終了である．

〔図1〕Konfabulatorを起動したところ

DATA

名称：Konfabulator
作者：Arlo Rose氏，Perry Clarke氏
ウェブサイト：http://www.konfabulator.com/
現在のバージョン：1.0.2
ダウンロードサイズ：3.7Mバイト
OS：MacOS(Mac OS X 10.2以降)
価格：$25

Konfabulatorを起動すると，**図1**のように，いくつかの特徴的なウィンドウが表示される．これらのウィンドウがそれぞれ，一つのWidgetによるものとなっている．Konfabulatorには，標準で10個のWidgetが付属する(**表1**)．これらは，Konfabulatorが最初に起動されたときに，ユーザーのホームディレクトリにある「書類」フォルダ内に，「Widgets」というフォルダが作られ，その中にコピーされる(**図2**)．

Konfabulatorは通常のアプリケーションと異なり，起動中Dock[注1]上にアイコンが表示されない．その代わり，メニューバーの右側に，Konfabulatorを表す歯車型のメニューが表示され，そこから操作を行う(**図3**)．Widgetの起動や，Konfabulator自体の終了や設定変更などは，このメニューから行うことになる．

〔表1〕標準Widgetの一覧

AirPort Signal	現在のAirMac(AirPort)のシグナルの強さを表示
Analog Clock	アナログ時計
Battery	現在のバッテリの残量を表示
Calendar	カレンダを表示
iTunes Remote	iTunes(Apple純正MP3プレーヤ)をコントロールするWidget
Picture Frame	画像フォルダ内の画像をスライドショー表示
Stock Ticker	株価情報を表示
Digital Clock	デジタル時計
The Weather	指定した都市の天気，気温情報を表示
What To Do?	ToDoリスト

注1：Dock――Mac OS Xに標準搭載されているランチャ兼プロセスビューワ．Windowsのタスクトレイにランチャ機能を付加したようなもの．

〔図2〕
標準でインストールされる10個のWidget

〔図3〕Konfabulatorのメニュー

Widgetの起動は，Widgetのアイコンを直接ダブルクリックすることでも行うことができる．各Widgetは，通常一つのウィンドウをもち，その機能はWidgetによってさまざまだが，コンテクストメニュー注2を表示すると，そのWidgetのアバウト画面や設定画面，Widgetの終了などを行うことができる．

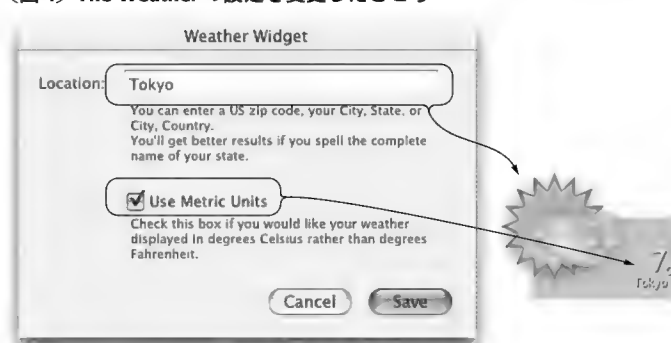
設定画面において何が設定できるのかは，Widgetによって異なる．たとえば，標準のWidgetであるThe Weatherは，現在の天気と気温を表示するWidgetだが，設定画面で表示する都市名，および気温を華氏と摂氏どちらで表示するのかを指定できる(**図4**)．

The Weatherは，指定された都市名をもとに，weather.comというWebサイトにアクセスして，天気情報を取得するため，このサイトに登録されていない都市名を指定しても，正しく情報を取ることができないが，たとえばTokyoを指定すれば，東京の現在の天気と気温を知ることができる．また，「Use Metric Units」というチェックボックスは，気温の表示を華氏から摂氏に変換するための設定である．weather.comはアメリカのサイトであるため，気温はアメリカ式に華氏で表示されているが，このチェックボックスにチェックを入れると，Widget内で華氏→摂氏の変換が行われ，摂氏で表示されるようになるのだ．

Widgetの設定は，一度設定を行えばWidgetを終了しても保持される．また，Konfabulator自体も，終了時に起動していたWidgetの種類や位置を記憶しており，再度立ち上げれば，終了時の状態を復元できるようになっている．

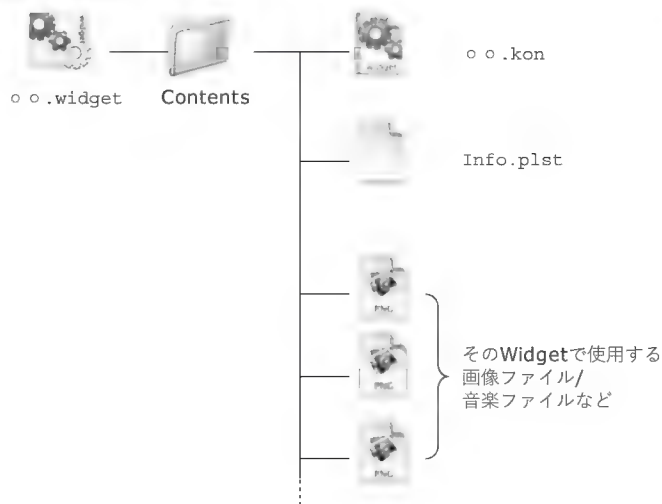
Widgetは，XMLとJavaScriptで作られている．詳細は後述するが，その簡単なしくみのため，Widgetの改造や作成も比較的容易である．そして，Konfabulatorのサイトにはガallery Galleryという，ユーザーが，自分が作ったWidgetを登録して公開できるスペースが用意されており，数多くのWidgetが登録され，その数は日々増えつづけている．その中から，自分の目的に合った，もしくは自分の好きなWidgetを自由にダウンロードして，自分にとって便利なデスクトップ環境を構築することができるようになっているのだ．

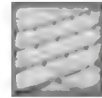
注2：コンテクストメニュー——Macの一つボタンマウスの場合はCtrl+クリックで表示される．2ボタンマウスを利用していれば，右クリックもできる．

〔図4〕The Weatherの設定を変更したところ

Column 1 Widgetがたくさん登録されたGallery

KonfabulatorのWebサイトには，「Gallery」というコーナーがあり，そこには世界中のKonfabulatorユーザーが作成したWidgetが登録されている(**図A**)．標準のWidgetの機能を強化したものから，ゲームや，何の機能ももたず目で見て楽しむ環境系Widgetまで，その種類はさまざまである．これらのWidgetから，自分に必要なものを選んで利用したり，自分でWidgetを作成する際の参考にしたりすることができる．

〔図A〕konfabulator.comのGallery

〔図5〕Widgetの中身

〔リスト1〕時計を表示するWidgetの.konファイル

```
<?xml version="1.0" encoding="UTF-8"?>
<widget version=1.0>

    <window title="Clock">                          ┐
         <name>main_window</name>                   │
         <width>100</width>                         │
         <height>18</height>                        │
         <opacity>100%</opacity>                    ├ Widget全体の定義
         <visible>1</visible>                       │
         <shadow>0</shadow>                         │
    </window>                                       ┘

    <image src="Background.png">                    ┐
         <name>background</name>                    │
         <hOffset>0</hOffset>                       │
         <vOffset>0</vOffset>                       │
    </image>                                        │
                                                    │
    <text>                                          ├ Widgetの画面上の
         <name>clock</name>                         │  構成物の定義
         <font>Trebuchet MS</font>                  │
         <size>12</size>                            │
         <color>#ffffff</color>                     │
         <hOffset>14</hOffset>                      │
         <vOffset>50</vOffset>                      │
         <alignment>center</alignment>              │
    </text>                                         ┘

    <action trigger="onLoad">                       ┐
                                                    │
         function updateTime() {                    │
              theDate = new Date();                 │
              theHour = String(theDate.getHours()); │ Widget
              theMinutes = String(theDate.getMinutes()); ├ の動作の
              theSeconds = theDate.getSeconds();    │ 定義
              clock.data = theHour +                │
":"+theMinutes+":"+theSeconds;                      │
         }                                          │
         updateTime();                              │
    </action>                                       ┘

    <action trigger="onTimer" interval="1">
         updateTime();
    </action>

    <about-box>                                     ┐
         <image>About.png</image>                   ├ アバウトダイアログの定義
    </about-box>                                    ┘

</widget>
```

Widgetのしくみ

● Widgetのファイル構成

Widgetはアプリケーションバンドル，という形式になっている．これは，フォルダをまるで一つのファイルのように見せかけるしくみである．したがって，Widgetは見かけはファイルのようだが，じつは中にいくつものファイルを内包したフォルダになっている．その中身を見るには，コンテクストメニューを開き，「パッケージの内容を表示」を選択する．

Widgetの内部のファイル構成は，**図5**のようになっている．必ず必要なのは，「.kon」という拡張子をもつファイルで，これがWidgetのプログラム本体である．「info.plst」はplst(プロパティリスト)形式という，Appleが情報を格納するために一般的に利用しているXMLのデータ形式で書かれたバージョン情報である．このファイルは必須ではないが，このファイルが存在することで，Finder上でWidgetの情報を表示した際に，バージョン情報が表示されるようになる．

それ以外のファイルは，Widgetが利用する画像ファイルなど，個々のWidgetごとに，必要となるファイルである．

● .konファイルの形式

先に述べたとおり，Widgetのプログラム本体は.konファイルである．このファイルはXMLとJavaScriptで構成されており，XMLでWidgetのデザインや属性を，JavaScriptでその挙動を記述するようになっている．

簡単なサンプルを**リスト1**に示す．これは，画面上に時計を表示する簡単なWidgetである．

Widgetの定義は，まずWidget全体を表すウィンドウを定義することからはじめる．Konfabulatorでは，まずウィンドウを一つ定義し，そこに画像やテキストを配置していく，という方法で画面を構築するのである．ウィンドウの定義には，window要素を利用する．window要素では，ウィンドウの位置や大きさ，透明度など，Widget全体の属性を設定することになる．window要素で設定可能な属性値を**表2**に示す．たとえば，次のように設定した場合は，300×200のサイズのウィンドウで，透明度が50%，常に前面に表示されるWidgetが生成される．

〔表2〕window要素で定義できる属性値

height	ウィンドウの高さ
hOffset	ウィンドウの位置(Y座標)
level	フローティングウィンドウなどの設定
name	ウィンドウの名前
opacity	ウィンドウの透過度
shadow	影をつけるかどうか(1：影をつける)
title	ウィンドウのタイトル
visible	ウィンドウが可視かどうか(1：可視)
vOffset	ウィンドウの位置(X座標)
width	ウィンドウの幅

```
<window>
    <title>MyWidget</title>
    <name>main_window</name>
    <width>300</width>
    <height>200</height>
    <alignment>left</alignment>
    <opacity>50%</opacity>
    <shadow>1</shadow>
    <visible>1</visible>
    <level>topMost</level>
</window>
```

なお，各要素の属性値は上記のように，window要素の内部に独自の要素として定義できるほか，次のように定義することも可能である．

```
<window title="MyWidget" name="main_window"
width="300" height="200" alignment="left"
opacity="50%" shadow="1" visible="1"
level = "topMost" />
```

さて，window要素で定義されるウィンドウは，Widget全体の属性を設定するために用いられる仮想的なウィンドウであり，実際には画面に表示されない，いわば透明なウィンドウである．各Widgetでは，window要素で定義された領域内に画像やテキストを配置することで，画面を作成する(**図6**)．画像とテキストの配置に利用するのは，image，およびtext要素である．

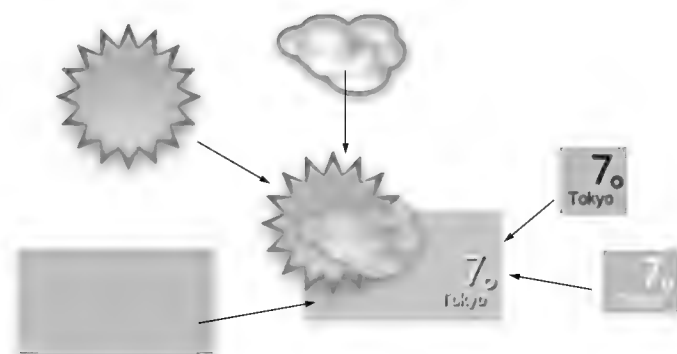
画像は，image要素を用いて指定する．実際の画像データは，src属性においてファイル名で指定する．通常はWidget内に内包した画像を指定するが，ファイルパスさえ間違わずに指定すれば，外部にある画像を指定することも可能である．利用可能な画像のフォーマットはドキュメントではとくに触れられてはいないが，PNG，JPEG，GIFなどが利用可能である．標準Widgetでは，PNGが多く使われているが，これはPNGにアルファチャネルの機能があり，画像の透過度を自在に調節できるためだと思われる．

```
<image src="Background.png">
    <name>background</name>
    <hOffset>0</hOffset>
    <vOffset>0</vOffset>
</image>
```

image要素では，画像の透過度，大きさのほか，回転角度の指定を行うことができる．標準WidgetであるAnalog Clockでは，この機能を利用して，時計の針の表示を行っている．

画面上に文字を表示するにあたっては，text要素を利用する．こちらは，表示する文字列やその位置，フォントなどを指定することができる．

```
<text data="Unknown">
    <name>myName</name>
    <font>Trebuchet MS</font>
```

〔図6〕Weatherを例にしたWidgetの画面の構成

```
    <size>12</size>
    <color>#ffffff</color>
    <hOffset>164</hOffset>
    <vOffset>110</vOffset>
</text>
```

● 設定画面

Konfabulatorではpreferenceという要素を利用することで，Widgetに設定画面を追加することができる．設定画面は，Widget上でコンテクストメニューを表示して，「Widget Preferences...」を選択すると，表示される独立したウィンドウで，テキストやチェックボックスを使って，Widgetの設定をユーザーが指定できるものだ．ここで指定された値は，JavaScriptから参照することができ，プログラム中で設定に応じた処理が可能になる．設定された値は，Widgetを終了しても保持される．

preference要素のサンプルを**図7**に示す．設定できるのは，JavaScriptから参照する際に利用する名称(name)，設定画面上で表示されるタイトル(title)と説明文(description)，設定項目のタイプ(type)と初期値(defaultvalue)である．また，隠し項目を設定できるhiddenという要素も指定できる．これを利用した場合は，設定画面にはその項目は現れない．これは，その項目をJavaScript中でのみ設定/参照を行う，静的変数的に利用する場合に用いる．

設定項目のタイプとして設定できるのは，ドキュメントによればtext(テキストボックス)，checkbox(チェックボックス)，popup(ポップアップメニュー)の三つである．それに加え，ドキュメントには書かれていないが，slider(スライダ)というタイプも指定できる．textとpopupの場合は文字列，checkboxは1と0，sliderは0～250の数値が保存できる．そして，popupの場合は，HTMLと同様にoption要素を使って，メニュー項目を指定できる．

JavaScriptから設定項目を参照する場合は，次のようにvalueという要素を利用する．

```
alert(preferences.dataname.value);
```

● JavaScriptの記述場所

Konfabulatorでは，JavaScriptはaction要素の中に記述す

る．action要素にはtriggerという属性を定義でき，ここでそのJavaScriptを呼び出すタイミングを指定できる．いわゆるイベント駆動型のプログラミングとなる．呼び出すことのできるタイミングは**表3**のとおりである．

```
<action trigger="onLoad">
    var count = 0;
    alert("起動されました！");
</action>
<action trigger="onTimer" interval="10">
    count++;
</action>
```

また，imageおよびtext要素では，onMouseEnter，onMouseExit，onMouseDown，onMouseUpの四つの要素が定義可能で，この中にもやはりJavaScriptを記述できる．これを利用すると，その画像や文字の上で，マウスがクリックされたときにJavaScriptを実行する，といったことが可能になる．

```
<image src="button.png">
    <name>background</name>
    <hOffset>20</hOffset>
    <vOffset>50</vOffset>
    <onMouseUp>
    alert("クリックされました！");
    </onMouseUp>
</image>
```

KonfabulatorのJavaScriptエンジンは，JavaScript 1.5 standard（ECMA-262, revision 3[注3]）に対応している．したがって，この仕様で定義されている基本的なJavaScriptの構文や，Date，Arrayといった標準のオブジェクトはすべて利用することができる．

また，XMLで定義している画面上のオブジェクト，たとえばwindowやimage，textといった要素には，その要素内でname属性で定義した名前を用いてアクセスすることができる．たとえば，以下のようにすることで，「background」という名前のimage要素で表示する，画像ファイルを指定することができる．

```
background.src = 'background.png';
```

● Konfabulator独自のJavaScriptの命令/オブジェクトについて

Konfabulatorは，Mac OS Xに特化したソフトウェアであり，JavaScriptにも，Mac OS Xならではの命令が拡張されている．

〔図7〕preferenceのサンプル

(a) 設定画面

```
// 文字入力
<preference name="TextPref">
    <title>文字:</title>
    <type>text</type>
    <defaultValue>文字列入力欄</defaultValue>
    <description>好きな文字を入力してください．</description>
</preference>
// チェックボックス
<preference name="CheckPref">
    <title>チェック:</title>
    <type>checkbox</type>
    <defaultValue>0</defaultValue>
    <description>二者択一</description>
</preference>
// ポップアップメニュー
<preference name="PopPref">
    <title>ポップアップ:</title>
    <type>popup</type>
    <option>Apple</option>
    <option>Ogrange</option>
    <option>Lemon</option>
    <defaultValue>Ogrange</defaultValue>
    <description>選択肢</description>
</preference>
// スライダー
<preference name="SliderPref">
    <title>スライダー:</title>
    <type>slider</type>
    <defaultValue>50</defaultValue>
    <description>0～250の値</description>
</preference>
// 隠しフィールド
<preference name="TextPref">
    <hidden>true</hidden>
    <type>text</type>
    <defaultValue>これは見えません</defaultValue>
</preference>
```

(b) サンプルコード

〔表3〕JavaScriptを呼び出すことができるタイミング

トリガ名	呼び出されるタイミング
onGainFocus	Widgetがアクティブになったとき
onIdle	1秒に5回呼び出される
onLoad	Widgetが起動されたとき
onLoseFocus	Widgetがアクティブではなくなったとき
onMouseDown	マウスボタンが押されたとき
onMouseEnter	マウスカーソルがWidgetの中に入ったとき
onMouseExit	マウスカーソルがWidgetの外に出たとき
onMouseUp	マウスボタンが離されたとき
onPreferencesChanged	設定値が変更されたとき
onTimer	定期的に呼び出す（タイミングはinterval属性で秒単位で指定）
onUnload	Widgetが終了したとき
onWakeFromSleep	スリープ状態から復帰したとき
onWillChange Preferences	設定画面が表示される直前

注3：http://www.ecma-international.org/publications/standards/ECMA-262.htm

たとえばappleScriptは，AppleScriptを実行する命令である．

```
appleScript('tell application
            "Finder" to close every window');
```

これは，Finderに，開いているウィンドウをすべて閉じるように命令するスクリプトを実行する．このようにAppleScriptは，MacOSにおいて作業の自動化やアプリケーション間通信を行うためのスクリプト言語で，これを利用することで，他のアプリケーションを操作することができるものだ．標準Widgetである iTunesXXXは，AppleScriptを利用して，iTunesを操作している．

ほかにも，runCommandというUNIXのシェルコマンドを実行する命令や，マシンのバッテリの残量を調べることができるsystem.batteryオブジェクト，AirMac[注4]の状況を調べることができるsystem.airpotオブジェクトなど，Mac OS Xならではの命令が存在し，OSの機能を活用したWidgetの作成が可能となっている．

Konfabulatorの問題点

Konfabulatorは，まだまだ登場したばかりの新しいソフトウェアである．そのためもあり，使用しているといくつかの問題点もあることに気づく．

まず気になるのは，メモリを大量に消費する点である．Konfabulatorでは，各Widgetがそれぞれ，別のプロセスとして起動される．そして，それぞれのWidgetが10Mバイトほどのメモリを消費するのである．そのため，メモリを多く積んでいないマシンでは，立ち上げられるWidgetの数が限られてしまうし，多くのメモリを積んでいても，他のソフトウェアに割くことのできるメモリがWidgetにとられてしまうことに変わりはない．Konfabulatorは，マシンの起動とともに立ち上げておき，バックグラウンドでずっと立ち上げておきたいソフトウェアだが，メモリ消費量が大きいため，Widgetをいくつ立ち上げておくか，悩むところである．

また，すでに述べたとおりKonfabulatorでは，JavaScriptに独自の拡張をしている．しかし，その拡張はまだ，やや中途半端な印象を受ける．標準のWidgetとして，BatteryやAirport Meterなど，それらの拡張機能を使うWidgetがいくつか付属しているが，じつはOSの機能に直接アクセスする命令は，それらのWidgetで使われているものでほぼすべてともいってよい．

ほかにも画面サイズや色深度の取得や，クリップボードへのアクセスなど，多少の命令は用意されているものの，まだまだできないことのほうが多い．今後のバージョンアップにともなって，JavaScriptの仕様も拡張されていくのだろうが，現時点では，標準となるWidgetを作るにあたって，インパクトがあるものを作るにはどういう機能をつければいいか，を選択基準に命令を追加していったような印象を受けなくもない．

Column 2 ほかのプログラミング言語との連携

本文中でもふれたが，runCommandという命令を使うと，シェルコマンドを実行することができる．これを利用すると，現在Konfabulatorではサポートされていない機能，たとえば文字コードの変換や，http以外を利用したインターネットへのアクセス等を実現することも可能である．実際，Galleryに登録されているWidgetにも，runCommandを活用しているものが多く見られる．

Widgetはアプリケーションバンドルで，その内部にはどんなファイルでも含むことができるので，PerlやRubyなど，Mac OS Xが標準でサポートしているスクリプトなどを入れておき，それをJavaScriptから呼び出すことで，事実上どんな処理を行うWidgetでも作ることができるようになる．

また，日本語を扱うわれわれにはとくに気になる問題として，文字コードの変換を行う機能がないことがあげられる．日本語の表示自体は問題がないのだが，文字コードとしては，シフトJISしか表示することができない．

Widgetの有効な利用方法の一つに，Webページのデータをダウンロードして，そこから情報をピックアップして表示する，というものがある．しかし，日本語で記述されたWebページには，シフトJIS以外にもEUC-JPやISO-2022-JPなど，さまざまな文字コードが使われている．そのうち，シフトJIS以外の文字コードで書かれているものについては，runCommandなどでPerlなど他の言語で書かれた文字コード変換ルーチンを経由するなどの工夫をしないと，表示することができないのである．

このように，気になる点も多いKonfabulatorではあるが，まだ公開されてから日が浅いソフトウェアである．今後，これらの欠点も，徐々に改善されていくことを期待したい．

Widgetの開発環境

Widgetには，開発時に利用可能なデバッグ環境が用意されている．デバッグモードにするには，.konファイル内で，debug要素を利用する．

```
<debug>on</debug>
```

そして，Widgetを立ち上げると，デバッグ用のウィンドウが立ち上がり，そこに発生したイベントやエラーメッセージなどが表示されるようになる（**図8**）．また，JavaScript中でprint文を利用することにより，デバッグウィンドウ上に任意の文字

注4：Appleの無線LAN機器の名称．アメリカではAirPortという名前だが，日本では既存の登録商標との関係でAirMacという名前で販売されている．

〔図8〕
デバッグウィンドウ

を表示することができる．

ちなみに現在，Widgetを開発するための専用のツールは存在していない．ここまで解説したとおり，Widgetの本体はテキストファイルなので，作成にあたってはテキストエディタなどを使えばよく，とくに開発ツールは必要ではないが，その必要性はKonfabulatorの作者も認識しているようで，Konfabulatorのサイトの設置されたフォーラム(掲示板)において，今後開発する予定があるという発言がなされている．

Konfabulatorはなぜ注目されるのか？

Konfabulatorは，2003年2月の公開以来，Mac関係のコミュニティにおいて，非常に大きな注目を集めている．その理由としてまずあげられるのは，数多くのWidgetが公開されており，組み合わせて使うことで，自分独自の環境を作り上げられることや，WidgetがXMLとJavaScriptで構成されていて，簡単に改造や作成が可能である点があげられる．**リスト1**でも示した.konファイルの見た目は，Webページを構成するHTMLファイルともよく似ており，これまでプログラミングをあまり経験していない人にとっても，それほど抵抗感なく受け入れることができるものとなっているからだ．

ただ，Konfabulatorの機能は，XMLとJavaScriptを解釈して，それを実行する，ただそれだけのものでしかない，という意見もある．たとえば，Konfabulatorのサイトの設置されたフォーラム(掲示板)には，Konfabulatorが公開されてまだ間もないころに，「Konfabulatorは単なるXMLとJavaScriptのパーサである．これで$25は高すぎるのではないか」というメッセージが書き込まれている．

しかも，XMLとJavaScriptを使って，アプリケーションを作成できる，というシステムは，Konfabulator独自のアイデアではない．たとえば，オープンソースのWebブラウザであるMozillaには，XUL(XML-based User-interface Language)というインターフェース記述用の言語が搭載されている．これはXMLとCSS[注5]でインターフェースを，JavaScriptで動きをつけることができるもので，Mozillaのインターフェースはすべてこの XULで記述されているほか，それ以外のアプリケーションをXULを使って作成することも可能になっている．XULでは，残念ながら，それを使ったアプリケーションを作成する，という行為はあまり一般的なものとはなっていないが，XULとKonfabulatorは，XMLとJavaScriptを利用する，という点において非常によく似ており，筆者もKonfabulatorのことをはじめて知ったとき，まずXULのことを連想した．

しかし，単なるXML＋JavaScriptのパーサであったとしても，しかもXULがそれほど一般的にならなかったにも関わらず，Konfabulatorの登場は，Macコミュニティに大きなインパクトを与えている．これはなぜか？

Konfabulatorがこれほど注目された理由には，XMLとJavaScriptを使う，という特性以前に，その「見た目」が大きく影響しているように思う．Konfabulatorにははじめからいくつかの Widgetがついてくるが，それらのWidgetは，どれもみな，見た目を非常に重要視して作られており，たいへん美しいものになっている．Konfabulatorを初めて立ち上げたとき，それらのWidgetが次々に立ち上がるようすは，なかなか格好良く，印象的な光景である．そして続いてKonfabulatorを操作していくうち，それらの美しいWidgetが，簡単なしくみで作られており，自分でも簡単に改造したり，作成したりすることが可能であることがわかってくると，そこにさらに「何か自分でも作れるのではないか」という気分も加わり，Konfabulatorの世界にぐいっと引き込まれる，というわけだ．

作者のArlo Rose氏は，元Apple社のヒューマンインターフェイスデザインセンターに在籍したGUIエンジニアで，MacOS 9の時代にKaleidoscopeというツールを作成した人物である．こちらは，ウィンドウの外見や挙動を変更できるというデスクトップカスタマイズツールだったが，スキーマと呼ばれるデスクトップテーマの改造や作成が，簡単にできたこともあり，世界中のユーザーによってさまざまなスキームが公開されて，MacOSにおける「定番」アプリケーションとしての位置を獲得した．このKaleidoscopeを見ても，今回のKonfabulatorを見ても，技術云々だけではなく，それらをどういった形でソフトウェアにするか，という，「注目されるソフトウェア」を作成するためのデザインセンス，のようなものが感じられるように思う．

おわりに

繰り返しになるが，Konfabulatorはまだ新しいツールで，バージョンは1を超えてはいるものの，まだ発展途上にあるといってもよいものである．今後，どんな機能が搭載されていくのか，そして，どんな位置付けを獲得していくのか，今後も注目していきたい実行環境である．

みずの・たかあき

注5：CSS――カスケーディングスタイルシート．Webページなどのデザインを決めるための記述方式．

XScaleプロセッサ徹底活用研究

第1回　PXA25x/PXA26xアプリケーションプロセッサ解説

保坂一宏

CQ RISC評価キット/XScaleが発売された．本キットを活用するための，XScaleプロセッサについての解説記事の連載を今月号から開始する．最近の高機能化したPDAには，XScaleプロセッサが採用されている例が多い．今回は，XScaleプロセッサとはどんなCPUなのか，またCQ RISC評価キット/XScaleに搭載されているPXA250のアーキテクチャや内蔵機能など，ハードウェアの概要について解説する．（編集部）

はじめに

PXA25x/PXA26xは，インテルXScaleマイクロアーキテクチャを採用したアプリケーションプロセッサです．ハンドヘルド機器，ワイヤレス機器などの製品に適しています．2002年2月のPXA250の発表から1年以上が経過し，数々のPDAなどに実装された実績があります．

またこの3月には，新たにパフォーマンスが向上したPXA255，PXA260，PXA263が正式発表されました．PXA25x/PXA26xは，最高動作周波数200/300/400MHzの高性能で，低消費電力モードと細かな電力制御，マルチメディア機能の処理能力向上，豊富な周辺I/Oなどを特徴としています．

PXA25x/PXA26xに採用されているXScaleマイクロアーキテクチャは，ARMアーキテクチャV5TEに準拠しており，前身となるStrongARMとアプリケーションコードレベルで互換をもっています．また，スーパーパイプライン構造，40ビットアキュムレータ，SIMDなどにより，マルチメディア系の処理も高速に実行が可能です．PXA25x/PXA26xではさらに，各種OS，ミドルウェア，ライブラリ，ソフトウェア/ハードウェアの開発環境が整備されています．インテルからもインテグレーテッド・パフォーマンス・プリミティブ（IPP）ライブラリなどが供給されており，通信，信号処理，算術演算，メディア機能の最適化を図ることが可能です．

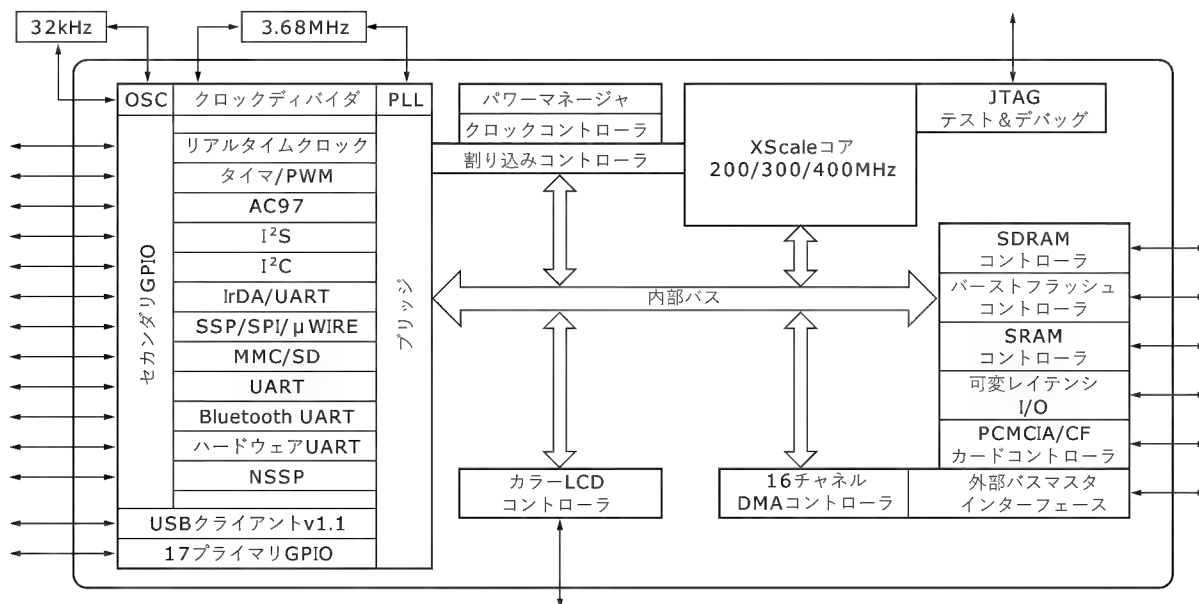
図1にPXA25x/PXA26xの内部構造を，**表1**に機能の一覧を示します．またPXA26xはインテルStrataFlashメモリがチップ内にスタックされて1チップとなり，さらにサイズも小さくなっています．**表2**にPXA26xのフラッシュ容量の違いを示します．

1　インテルXScale マイクロアーキテクチャ概要

XScaleコアの特徴を次に示します．

▶ ARM V5TEアーキテクチャ互換

XScaleコアは，ARM V5の整数演算命令をサポートしています．浮動小数点演算命令はハードウェアではサポートしていません．ARM VT5のTunmb命令，ARM V5EのDSP拡張命令をサポートしています．XScaleコアでは独自に16ビットSIMD

〔図1〕PXA255回路ブロック

〔表1〕PXA25x/PXA26xの機能一覧

CPUコア	XScaleマイクロアーキテクチャコア ARM V5TE，40ビットアキュムレータ，16ビットSIMD， 命令キャッシュ：32Kバイト，データキャッシュ：32Kバイト， ミニデータキャッシュ：2Kバイト，命令/データMMU， 分岐ターゲットバッファ
動作電圧	コア：0.85V～1.65V，I/O：3.3V，メモリ電源：2.5/3.3V
動作周波数	コアクロック：最大200MHz/300MHz/400MHz（ラインアップによる） 外部クロック：最大100MHz
パワーマネジメント	動作モード：RUN，TURBO，IDLE，SLEEP
外部メモリ	スタティックメモリ（ROM，フラッシュメモリ，SRAMなど）64Mバイト×6バンク SDRAM　64Mバイト×4バンク PCカード/コンパクトフラッシュ×2バンク
DMAコントローラ	16チャネル
LCDコントローラ	DSTN/TFTパネル対応　最大16ビットカラー　最大640×480ドット
割り込みコントローラ	FIQ/IRQ制御
タイマ	4本，ウォッチドッグタイマ設定可
リアルタイムクロック	アラーム割り込み　HZ割り込み設定可
シリアルインターフェース	UART　3チャネル（Full Function UART，Bluetooth UART，Standard UART） ハードウェアUART（PXA255/PXA26x）IrDA，I²C，I²S，AC97，SPI，SSP，NSSP（PXA255/PXA26x）
USBインターフェース	USB1.1対応　クライアント機能1チャネル
MMC/SDインターフェース	MMCモード，SPIモード転送に対応
GPIO	計81本（PXA250），85本（PXA255），90本（PXA26x）．他の機能とマルチプレクス，割り込み設定可
JTAG	5本，デバッグ機能あり
パッケージ	PXA25x：256ピンPBGAパッケージ（17×17/1mmピッチ） PXA26x：294ピンTPBGAパッケージ（13×13/0.65mmピッチ）

〔表2〕PXA26xシリーズのフラッシュ容量

プロセッサ	内蔵フラッシュメモリ	バス幅（ビット）	容量（バイト）
PXA260	なし	－	－
PXA261	128 K3 Intel StrataFlash memory × 1	16	16M
PXA262	128 K3 Intel StrataFlash memory × 2	16	32M
PXA263	128 K3 Intel StrataFlash memory × 2	32	32M

に対応したMAC機能を追加しています．インテルでは，これをメディアプロセッシングテクノロジと呼んでいます．

▶MACユニット

DSPコプロセッサ（CP0）が追加されています．このコプロセッサ0は40ビットアキュムレータと新規のMAC命令から構成されています．16ビットSIMDに対応しており，1命令で16ビットデータを複数扱うことができ，積和演算を高速に実行できます．

▶MMU

ARMアーキテクチャに準拠したMMUをもっています．またARMアーキテクチャと比較して新しいページ属性が追加されています．アクセス保護，論理アドレスから物理アドレスへの変換，キャッシュ制御を行うことが可能です．

▶命令キャッシュ

32Kバイト，32ウェイセットアソシアティブ方式の命令キャッシュをもっています．ラインサイズは32バイトです．またクリティカルなコードのキャッシュロックが可能です．

▶分岐ターゲットバッファ

分岐予測のための128エントリのダイレクトマッピングのキャッシュです．分岐系の命令でのペナルティを少なくします．

▶データキャッシュ

32Kバイト，32ウェイセットアソシアティブ方式のデータキャッシュと2Kバイト，2ウェイセットアソシアティブ方式のミニデータキャッシュをもっています．ラインサイズは32バイトです．ライトスルー，ライトバックの設定が可能です．PXA255/PXA26xではライトバック動作時の制限が修正されており，パフォーマンスが向上しています．

▶ライトバッファ/フィルバッファ

ライトバッファ/フィルバッファにより，XScaleコアからの外部メモリアクセス要求を効率的に行うことができます．

▶パフォーマンスモニタリング

デバッグ用にキャッシュ効率などのパフォーマンス測定が可能になっています．JTAGデバッガなどで使用できます．

▶パワーマネジメント

クロックと電源制御が可能です．詳細は後述します．

▶デバッグ

デバッグ用のブレークポイントレジスタ，ミニ命令キャッシュなどを備えています．JTAGポートを介してデバッガを接続可能です．

2 クロックおよび電源制御

● クロック

PXA25x/PXA26xでは，クロック入力として3.6864MHzと32.768KHzの発振子を使用します．内部にOSCを内蔵してお

り，発振子を直結するだけで発振が可能です．外付けのコンデンサも必要ありません．32.768KHzの発振子はリアルタイムクロックおよびパワーマネージャ用ですが，3.6864MHzクロックからも供給可能なため，低消費電力にこだわらないシステムやコスト重視のシステムでは必須ではありません．

PXA25x/PXA26xには，コアPLL，内部ペリフェラル用の95.85MHzと147.46MHzの三つのPLLが内蔵されています．コアPLLはCPUコア，メモリコントローラ，LCDコントローラ，DMAコントローラ用のクロックを供給します．CCCRレジスタのL，M，Nの各パラメータにより，メモリ周波数，RUNモード周波数，TURBOモード周波数が決定されます．それぞれのパラメータは，次の関係になっています．

メモリ周波数＝L × 3.6864MHz（L：27，32，36，40，45）
RUNモード周波数＝M ×（メモリ周波数）（M：1，2，4）
TURBOモード周波数＝N ×（RUNモード周波数）
（N：1，1.5，2，3）

PXA255/PXA26xでは設定可能なMパラメータとして4が追加されており，RUNモードで400MHzの動作が可能になっています．また，内部バスが200MHzで動作可能になっています．

● CPU動作モード

PXA25x/PXA26xには，電力制御を行うために動作モードとしてRUN，TURBO，IDLE，SLEEPの四つのモードがあります．これらのモードを有効に切り替えることにより，消費電力の低減が可能です．

▶ RUNモード

通常の動作モードです．リセット直後は，RUNモードで周波数は約100MHzとなっています．RUNモードの周波数設定も必要であれば変更します．

▶ TURBOモード

高速動作モードです．RUNモードの何倍かで動作するようにあらかじめ設定しておきます．RUNモードとの切り替えは，処理を止めることなく可能です．

▶ IDLEモード

CPUコアへのクロックが停止します．CPUコア以外へのクロックは供給されているため，割り込みにより素早い復帰が可能です．

▶ SLEEPモード

CPUコアへの電源供給が止まります．内蔵ペリフェラルもRTCとパワーマネージャ以外は停止します．リセット，RTCまたはGPIOからの起動イベントにより復帰します．

● リセット

リセットには，ハードウェアリセット，ウォッチドッグリセット，GPIOリセットがあります．

▶ ハードウェアリセット

nRESET入力ピンを“L”レベルにアサートすることによりハードウェアリセットがかかり，nRESET_OUT出力ピンが“L”レベルにアサートされます．ハードウェアリセットではすべての機能が初期化されます．nRESETピンのネゲートにより，ハードウェアリセット内部の3.6864MHz OSCとPLLが安定した後nRESET_OUTピンがネゲートされ，ブートシーケンスを開始します．

▶ ウォッチドッグリセット

プログラムの暴走を検出してリセットがかかります．内蔵のOSタイマをウォッチドッグタイマとして使用できます．クロックとパワーマネージャ以外のすべてのユニットにリセットがかかります．

▶ GPIOリセット

GP[1]ピンのアサートによりリセットをかけるように設定することが可能です．RTC，クロック，パワーマネージャ，メモリコントローラ以外のユニットがリセットされます．

● 電源ピン

PXA25x/PXA26xの電源には，次のものが必要です．

▶ V_{CC}（0.85V～1.65V）

内部ロジック用電源です．外部の電源ICから動作周波数により電圧を可変させることが可能です．PXA255/PXA26xでは400MHzでも1.3Vで動作できるように改良されており，低消費電力性能が向上しています．

▶ PLL_V_{CC}（0.85V～1.65V）

PLL用電源です．V_{CC}と同じ電圧を供給します．

▶ V_{CC}N（3.3V/2.5V）

外部メモリバス，PCMCIAピン用電源です．

▶ V_{CC}Q（3.3V）

外部メモリバス，PCMCIAピン以外のI/O用電源です．PXA26xでは2.775Vの設定も可能となっています．

PXA26xではそのほかに内蔵フラッシュメモリ用のコア電源，I/O電源があります．

3 メモリコントローラ

外部バス領域としては，SRAMなどのスタティックメモリエリア，SDRAM専用エリア，PCカード/コンパクトフラッシュ専用エリアがあります．**図2**にPXA25x/PXA26xのメモリマップを示します．

● スタティックメモリインターフェース

CS0～CS5のチップセレクトにおいてそれぞれ64Mバイトのメモリ空間があり，個別の設定が可能です．設定可能なメモリタイプは，次のものがあります．

- ノンバーストROM/フラッシュメモリ
- バーストROM/フラッシュメモリ
- SRAM
- 可変レイテンシI/O（VLIO）
- シンクロナススタティックメモリ（CS0～CS3のみ）

各チップセレクトにおいて16ビットまたは32ビットのバス幅の設定が可能です．また，内部レジスタによるアクセスタイム

〔図2〕PXA25x/PXA26xのメモリマップ

アドレス	領域
0xFFFFFFFF	予約
0xB0000000	SDRAM バンク SDCS3（64Mバイト）
0xAC000000	SDRAM バンク SDCS2（64Mバイト）
0xA8000000	SDRAM バンク SDCS1（64Mバイト）
0xA4000000	SDRAM バンク SDCS0（64Mバイト）
0xA0000000	予約
0x4C000000	内蔵レジスタ
0x40000000	PCMCIA/CF スロット0（256Mバイト）
0x30000000	PCMCIA/CF スロット1（256Mバイト）
0x20000000	予約
0x18000000	スタティックメモリエリア CS5（64Mバイト）
0x14000000	スタティックメモリエリア CS4（64Mバイト）
0x10000000	スタティックメモリエリア CS3（64Mバイト）
0x0C000000	スタティックメモリエリア CS2（64Mバイト）
0x08000000	スタティックメモリエリア CS1（64Mバイト）
0x04000000	スタティックメモリエリア CS0（64Mバイト） StrataFlash memory（PXA26x）
0x00000000	

の設定も可能で，最大で23メモリクロック分のウェイト設定が可能です．さらにアクセス速度の遅い，もしくはウェイト要求により応答速度の異なるデバイスを接続する場合は，VLIOの設定でレディ信号(RDY)によるウェイト制御が可能です．

アウトプットイネーブルにはnOEピンを使用します．ライトイネーブルにはSRAMなどの設定ではnWEピンが使用されますが，VLIOの設定時だけはnPWEピンが使用されるので注意が必要です．

SRAMおよびVLIOの設定においては，DQM[3：0]ピンにおいてバイトイネーブル制御が可能です．また，リセット直後はCS0エリアの0x0番地のリセットベクタから実行が開始されるので，CS0にはフラッシュメモリなどブート可能なデバイスが割り当てられていなければなりません．CS0のメモリ種別およびバス幅は，**表3**に示す外部端子BOOT_SEL[2..0]により決定されます．PXA26xの場合も，内蔵フラッシュに合った設定にしておく必要があります．

図3にバーストROMリードサイクルのアクセスタイミングを，**図4**にSRAMライトサイクルのアクセスタイミングを，**図5**にVLIOライトサイクルのアクセスタイミングを示します．

● SDRAMインターフェース

SDCS0～SDCS3でそれぞれ最大64Mバイトの容量までSDRAMを接続可能です．16ビットまたは32ビットのバス幅の構成が可能です．SDCS0/1のペアとSDCS2/3のペアで異なった設定をすることも可能です．設定できるSDRAMクロックの最大周波数は100MHzです．内部メモリ周波数が100MHzを超えた場合，SDRAMクロックは内部メモリ周波数の1/2にする必要があります．

〔表3〕BOOT_SELの定義

BOOT_SEL 2	1	0	ブートメモリの種類
0	0	0	非同期32ビット ROM
0	0	1	非同期16ビット ROM
0	1	0	予約
0	1	1	予約
1	0	0	32ビット同期マスクROM(64Mビット) 16ビット同期マスクROM(32Mビット)×2 ＝32ビット幅
1	0	1	16ビット同期マスクROM(64Mビット)
1	1	0	16ビット同期マスクROM(64Mビット)×2 ＝32ビット幅
1	1	1	16ビット同期マスクROM(32Mビット)

MDCNFGレジスタでバス幅，サイズ，CASレイテンシなどの設定を行い，MDREFRレジスタでリフレッシュ関連の設定を行います．MDMRSレジスタのセットでモードレジスタ設定サイクルが発生します．

図6に標準的なSDRAMアクセスタイミングを示します．

● 16ビットPCカード/コンパクトフラッシュインターフェース

PC Card Standard Volume2 Electrical Specification Release2.1およびCF+ and CompactFlash Specification Revision1.4に準拠した，PCカード/CFカード用のバスインターフェースをもっています．8ビット転送および16ビット転送をサポートしています．ただしCardBusはサポートしていません．

具体的な外部デバイス接続例として，SRAMやROMなどの32ビット外部メモリの接続例を**図7**に，16ビットI/Oデバイスの接続例を**図8**に示します．

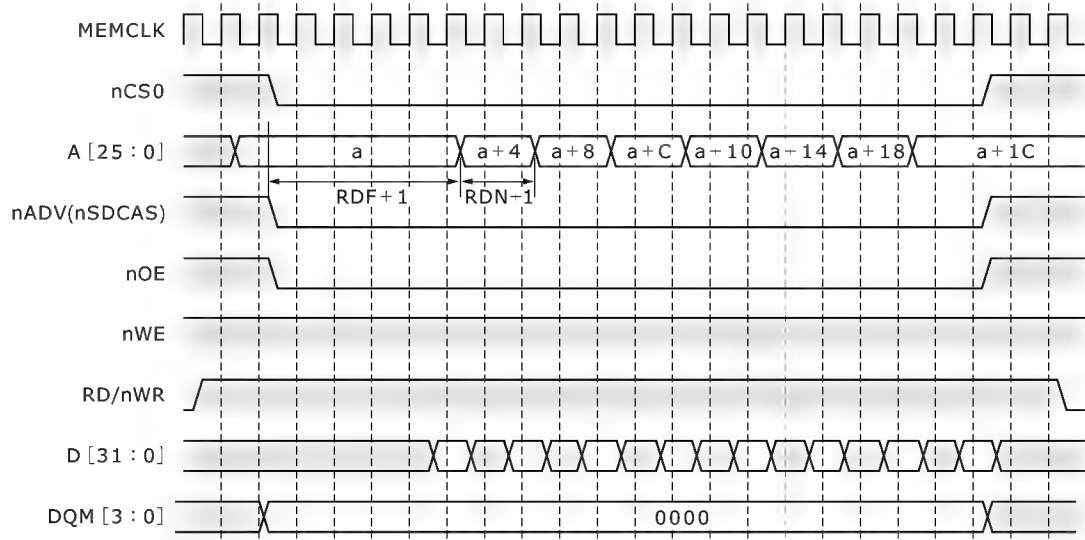
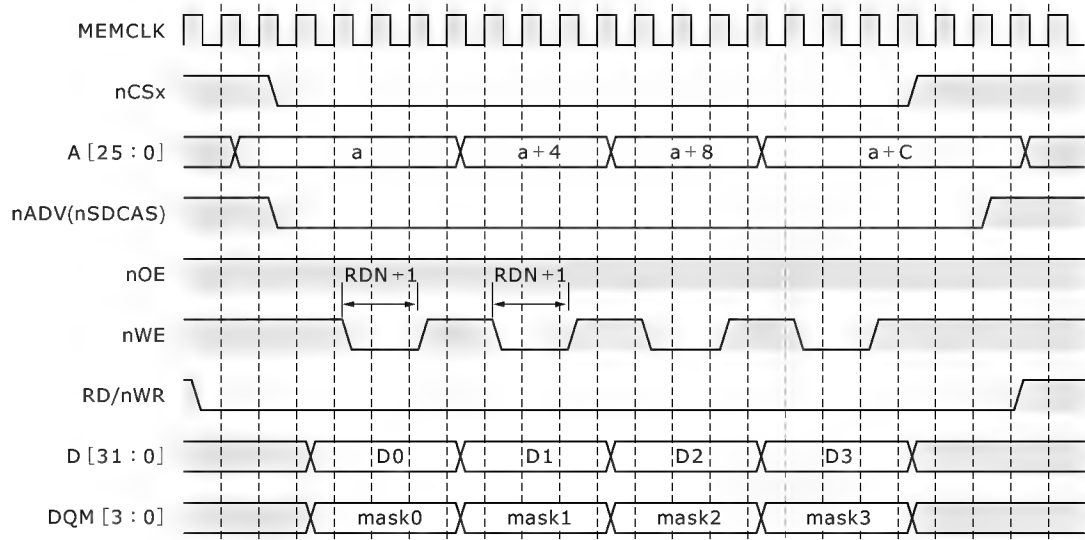
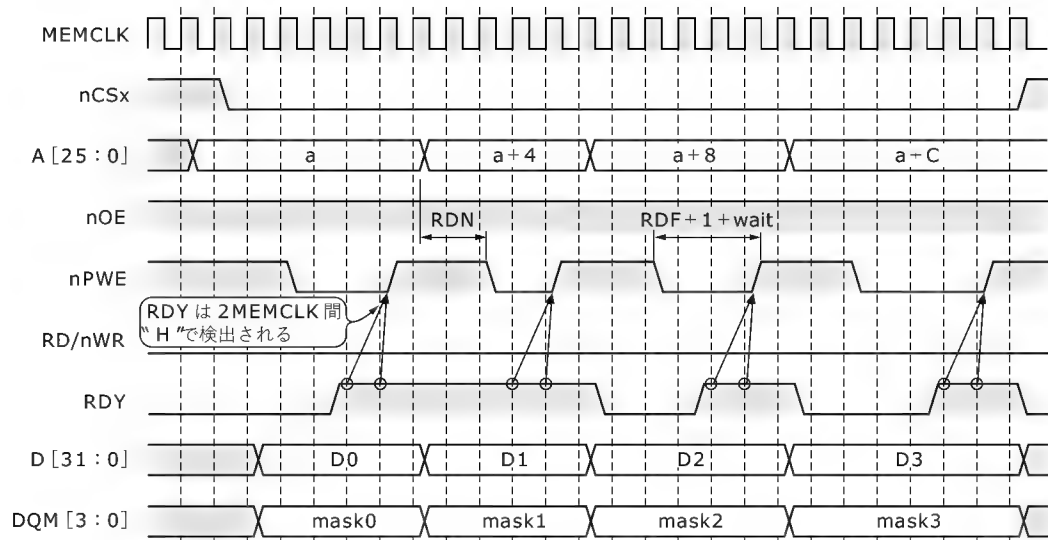
5 I/Oポート

PXA250では，I/OポートはGPIOとして合計81本あります．PXA255では85本，PXA26xではさらに追加されており，合計90本となります．ほとんどの信号は他の機能との兼用端子となっています．I/Oポートとして選択する際は，他の機能で使用しないことを確認する必要があります．

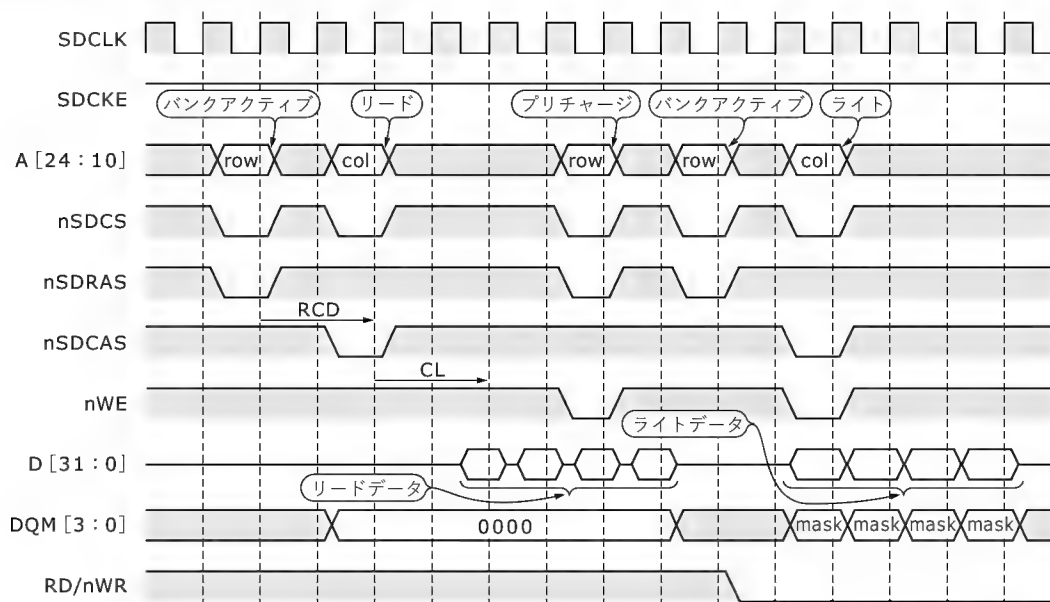
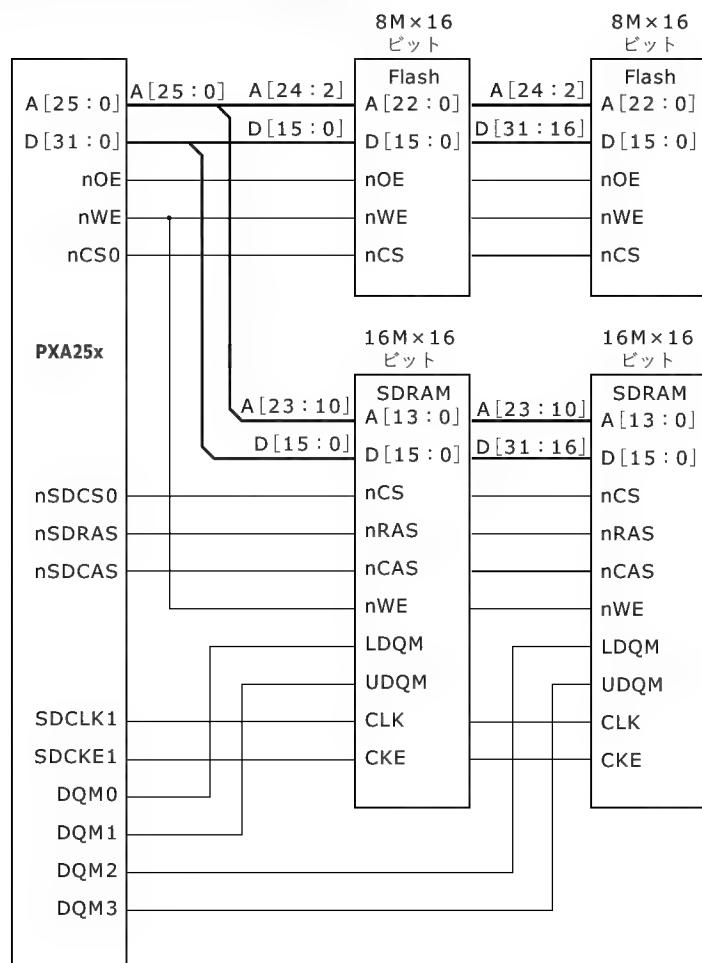
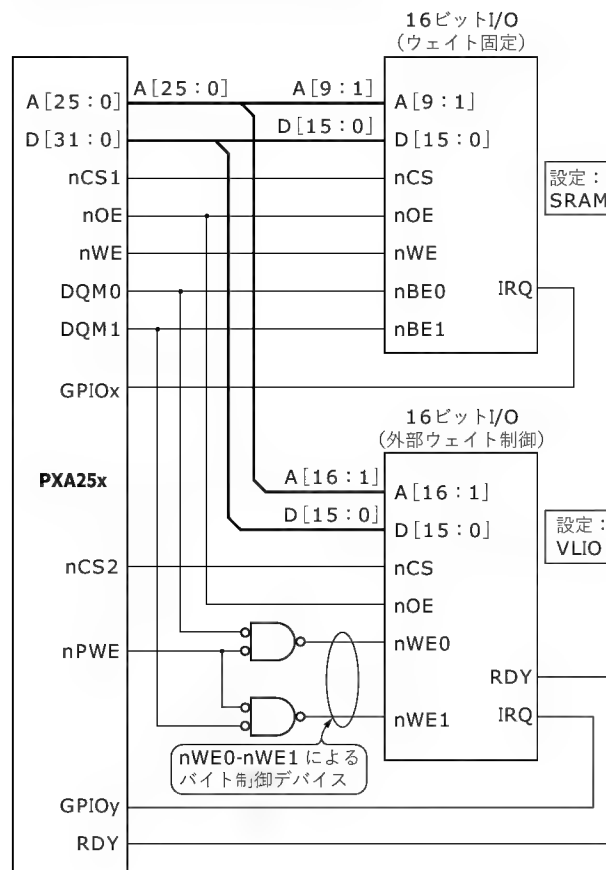
レジスタの設定により，方向制御，機能選択，レベルリード，出力時のセット/クリア，入力時のエッジ検出（立ち上り/立ち下り）が可能です．エッジ検出を割り込みコントローラで割り込み要因として使用することも可能です．

6 割り込み制御

割り込みには，FIQ割り込み，IRQ割り込み，ソフトウェア割り込みがあります．優先順位は高いほうからFIQ割り込み，

〔図3〕
32ビットバーストROM/フラッシュリードサイクルのアクセスタイミング

〔図4〕
32ビットSRAMライトサイクルのアクセスタイミング

〔図5〕
32ビットVLIOライトサイクルのアクセスタイミング

〔図6〕SDRAMのアクセスタイミング（CL＝2，RCD＝2）

〔図7〕32ビット外部メモリの接続例

PXA25x
A[25:0]
D[31:0]
nOE
nWE
nCS0
nSDCS0
nSDRAS
nSDCAS
SDCLK1
SDCKE1
DQM0
DQM1
DQM2
DQM3
8M×16ビット Flash
A[24:2]
D[15:0]
D[31:16]
A[22:0]
nCS
16M×16ビット SDRAM
A[23:10]
A[13:0]
nRAS
nCAS
LDQM
UDQM
CLK
CKE

〔図8〕16ビットI/Oデバイスの接続例

IRQ割り込み，ソフトウェア割り込みの順となります．割り込み要因は，各GPIO，内蔵ペリフェラルがあり，割り込みコントローラで制御されます．設定により個別に有効/無効，FIQ割り込みとIRQ割り込みに振り分けることが可能です．

▶ソフトウェア割り込み

ソフトウェア割り込み命令(SWI)の実行によりスーパバイザモードに移行し，ソフトウェア割り込みベクタから実行を開始します．

▶IRQ割り込み

IRQ割り込みが発生するとIRQモードに移行し，IRQ割り込みベクタから実行を開始します．ICPRレジスタで要因を特定し，割り込み要因をクリアします．

▶FIQ割り込み

FIQ割り込みが発生するとFIQモードに移行し，FIQ割り込みベクタから実行を開始します．ICPRレジスタで要因を特定し割り込み要因をクリアします．

7 DMAコントローラ

PXA25x/PXA26xは16本のDMAチャネルをもっています．それぞれ四つのレジスタセットで構成されています．それぞれのチャネルが内部/外部からのDMA要求を設定できます．メモリtoメモリの転送も可能です．

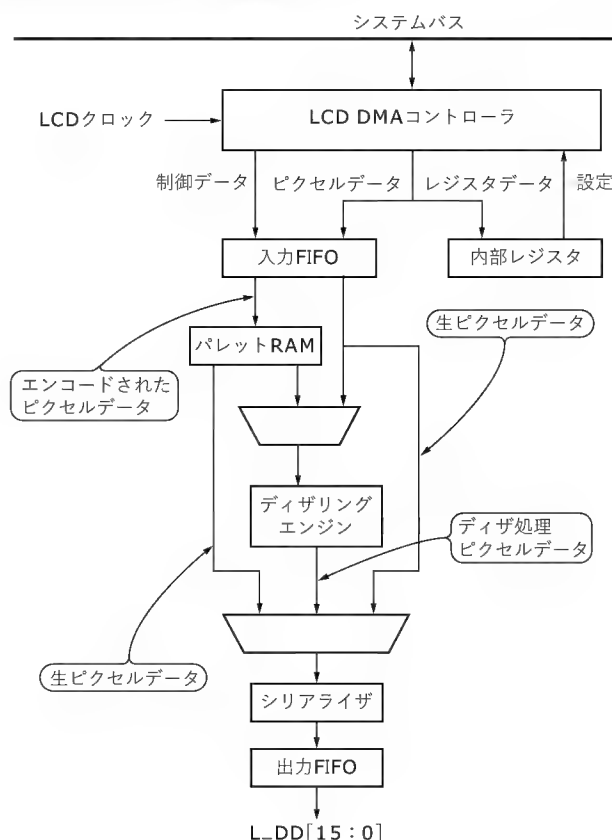
信号としては，外部ピンとして2本のDMAリクエスト(DREQ[1：0])と内部信号として内部ペリフェラルからの38本のリクエスト信号，割り込みコントローラへの割り込み信号があります．DREQを認識させるにはMEMCLKで4クロック以上アサートしておく必要があります．

各チャネルのレジスタでは，転送元/転送先アドレスのインクリメントあり/なし，転送幅，バーストサイズ，割り込みあり/なしなどの設定が可能です．最大転送長は8K－1バイトです．

16本のチャネルは4本ずつ4グループに分けられ，優先順位はチャネル0～3がいちばん高く，8回に4回サービスされます．以下チャネル4～7が2/8回，チャネル8～11/12～15が1/8回となっています．同一優先順位のチャネルはラウンドロビン式に実行されます．

またDMA動作モードとして，Descriptor Fetch ModeとNo-Descriptor Fetch Modeがあります．Descriptor Fetch Modeは，メモリ上に記述子を配置しておき，その記述子の内容がDMAのレジスタセットに転送されDMAを行います．

8 LCDコントローラ

PXA25x/PXA26xのLCDコントローラは，DSTN/TFTパネルに対応しています．また，シングル/デュアルディスプレイにも対応しています．8ビットまでのグレースケールモード，16ビットまでのカラーモードをサポートします．1024×1024までの液晶パネルサイズをサポートできますが，バスの占有率が高くなってしまうので，640×480より小さいサイズが推奨されています．

〔図9〕LCDコントローラ回路ブロック

表示用の画像データは外部メモリに格納され，専用DMAにより内蔵FIFOバッファへロードされます．FIFOバッファのピクセルデータは，パレットメモリに転送され，ディザリング回路を通過後，LCDのデータピンから出力されます．16ビットピクセルモードの場合は，パレットメモリはバイパスされ，直接LCDデータピンに送られます．LCDコントローラの回路ブロックを**図9**に示します．

9 シリアルインターフェース

PXA25x/26xには，各種シリアルコントローラが内蔵されています．

● 非同期シリアル(UART)

PXA25x/PXA26xは，Full Function UART(FFUART)，Bluetooth UART(BTUART)，Standard UART(STUART)の3本をもっています．レジスタセットはすべて同じで，機能は16550に準拠しています．**表4**にUARTの機能を示します．FFUARTはモデム用の制御線も含んだフル仕様，BTUARTはTxD/RxD/RTS/CTSの4本，STUARTはTxD/RxDの2本のみとなります．またFFUARTおよびBTUARTのRTS/CTSは

〔表4〕UART機能

キャラクタ長	5/6/7/8ビット
ストップビット	1/2(キャラクタ長6/7/8の場合), 1/1.5(キャラクタ長5の場合)
パリティ	なし/偶数/奇数
送受信バッファ	各64バイトFIFO
割り込み制御	受信ステータス割り込み/受信完了割り込み/送信要求割り込み/キャラクタタイムアウト割り込み/モデムステータス割り込み
ボーレート	BTUART, HWUART：最高921Kbps その他のUART：230Kbps
IrDA	低速IrDA転送対応
モデムコントロール	ソフトウェアフローコントロール, ハードウェアフローコントロール(HWUART)
自己診断機能	ループバックテスト, ブレーク/パリティ/フレーミングエラーシミュレーション

ハードウェアによる自動フロー制御には対応しておらず，ソフトウェアで制御しなければなりません．

● ハードウェアUART

PXA255/PXA26xでは上記の3本のUARTのほかに，ハードウェアフロー制御が可能なUART(HWUART)が追加されています．PCMCIAのピンまたは，BTUARTのピンと兼用になっています．

● 同期シリアル(SSP)

PXA25x/PXA26xの同期シリアルは，三つのフォーマットに対応しています．サポートするフォーマットを次に示します．

- ナショナルセミコンダクター　Microwire
- テキサスインスツルメンツ　Synchronous Serial Protocol (SSP)
- モトローラ　Serial Peripheral Interface (SPI)

● ネットワークSSP(NSSP)

PXA255/PXA26xでは，上記のSSPのほかにNSSPが追加されています．上記SSPの三つのフォーマットのほかに，プログラマブルシリアルプロトコルの設定が可能です．

● 赤外線通信ポート

内蔵の赤外線通信ポートは，Infrared Data Association(IrDA)の4Mbps仕様に対応しています．STUARTのピンとマルチプレクスされています．

● I²Cバスインターフェース

フィリップスが提唱するI²Cバスにも対応しています．SCL，SDAの2本の信号によりインターフェースされます．標準モードの100KbpsとFASTモードの400Kbpsに対応しています．

● AC97インターフェース

内蔵のAC97コントローラは，AC97 rev2.0仕様に準拠しており，AC-linkにより外部のAC97 CODECにインターフェースできます．

● I²S(Inter IC Sound)バスインターフェース

フィリップスのI²Sバスに対応しています．外部のI²S対応CODECにインターフェースできます．AC97のピンとマルチプレクスされています．

● USB

PXA25x/PXA26xは，USB 1.1に準拠したUSBデバイスコントローラ(UDC)をもっています．16本のエンドポイントをもち，フルスピードの12Mbpsに対応しています．バルク，コントロール，インタラプト，アイソクロナスの各転送に対応しています．PXA26xではシングルエンドでのインターフェースが，ほかの機能とマルチプレクスされて追加されています．

10 MMC/SDコントローラ

PXA25x/PXA26xのMMCコントローラは，The MultiMedia Card System Specification Version2.1に準拠しています．MMCおよびSDカードをサポートします．MMCモードやSPIモードの転送に対応しています．バス幅は1ビットのみで，SDカードの4ビット幅には対応していません．データ転送速度は最高20Mbpsです．MMC/SDインターフェースとして，MMCLK，MMCMD，MMDAT，MMCCSの4本の信号が用意されています．カード認識(CD)信号，ライトプロテクト(WP)信号，SDIOカードを使用する場合の割り込みは，別途GPIOを使用します．

11 CQ RISC評価キット/XScale CPUボードの仕様

● CPUボード概要

次に，CQ RISC評価キット/XScaleのCPUボードの仕様について解説します．**図10**にCPUボードのブロック図を，**写真1**にCPUボードの外観を示します．

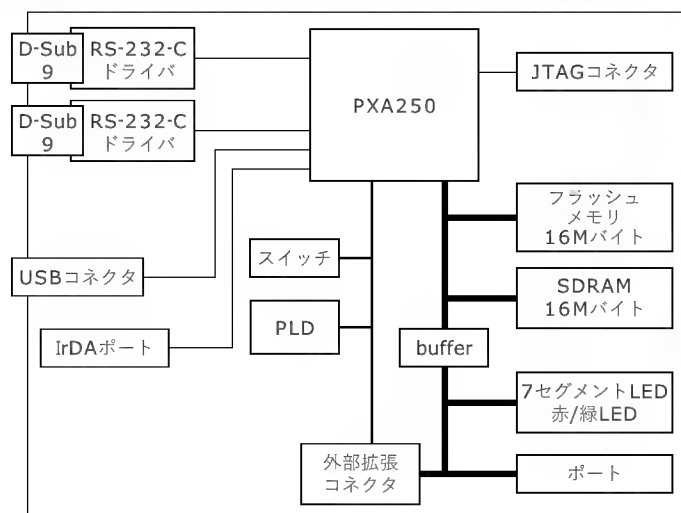
CPUボードには，内部最大400MHzで動作するPXA250を搭載しています．メモリはCS0の空間に16Mバイトのフラッシュメモリを，SDRAMバンク0に16MバイトのPC100仕様のSDRAMを実装しています．どちらも32ビットバス幅で接続しています．

CPUボードにはさらに，7セグメントLEDを一つ，LEDを二つ，8ビットディップスイッチを一つ(うちユーザー開放は6ビット)，プッシュスイッチを三つ(うち一つはリセットスイッチ)があるので，スイッチ入力やLED点灯出力などはすぐに試してみることができます．

インターフェースとしては，D-Sub9ピンのシリアルコネクタを二つ，それぞれCPUのFFUARTとBTUARTに接続されています．またIrDAトランシーバモジュールも実装されてるので，ソフトウェアを用意すればIrDAの通信も可能です．もう一つ，PXA250はUSBデバイスコントローラを内蔵しているので，パソコンなどのUSBホストと接続して，USBターゲット機器を実現することもできます．

またCPUのローカルバスに何らかのデバイスを接続しての評価もできるように，CPUボード上にはCPUのほぼすべての信号線をスルーホールまで配線し，基板の外に引き出せるようになっています．アドレスバスやデータバスは，バスバッファを経由

〔図10〕CQ RISC評価キット/XScale CPUボードのブロック図

〔写真1〕CQ RISC評価キット/XScale CPUボードの外観

して出力しているので，外部に5V電源系デバイスを接続することもできます．

● デバッガ接続インターフェースとしてシリアル接続とUSB接続に対応

CQ RISC評価キット/XScaleのデバッガは，基本的にリモートモニタ型デバッガです．デバッガUIを走らせるホストパソコンとは，何らかのインターフェースを使って接続する必要があります．

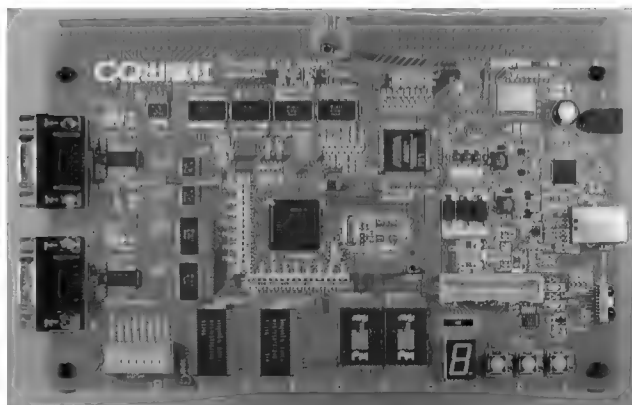
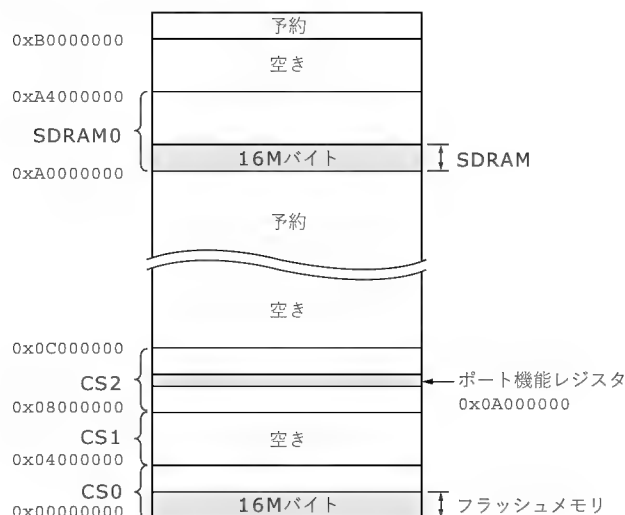
CQ RISC評価キット/XScaleでは，リモートモニタ型デバッガで一般的に使われるRS-232-Cによるシリアル接続以外に，PXA250内蔵のUSBデバイスコントローラの機能を使って，USBでの接続も可能になっています．これによりユーザープログラムの高速なダウンロードや，レスポンスのよいシングルステップ実行などが実現できます．

● CPUボードメモリマップ

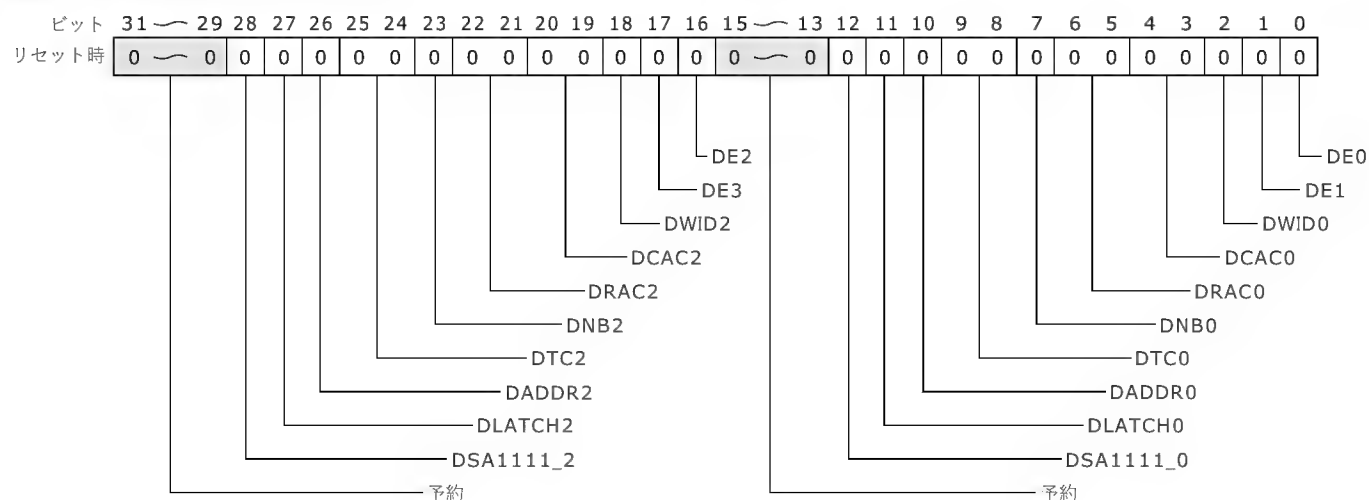
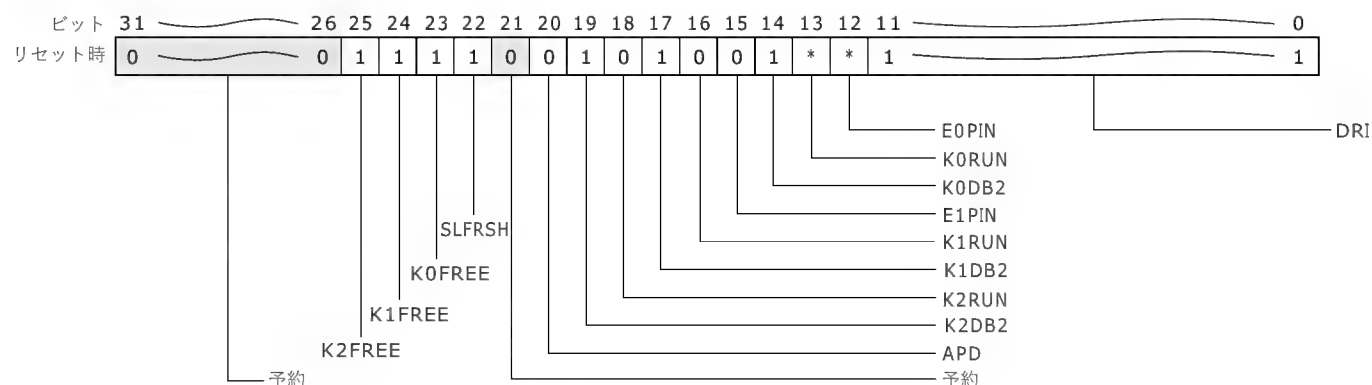
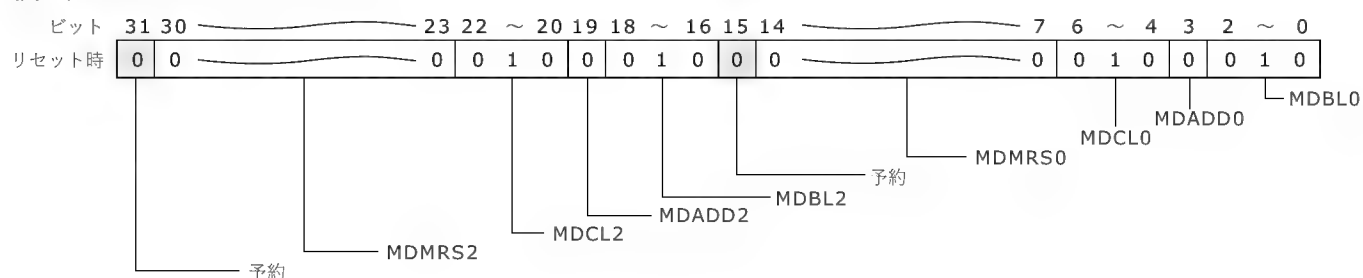
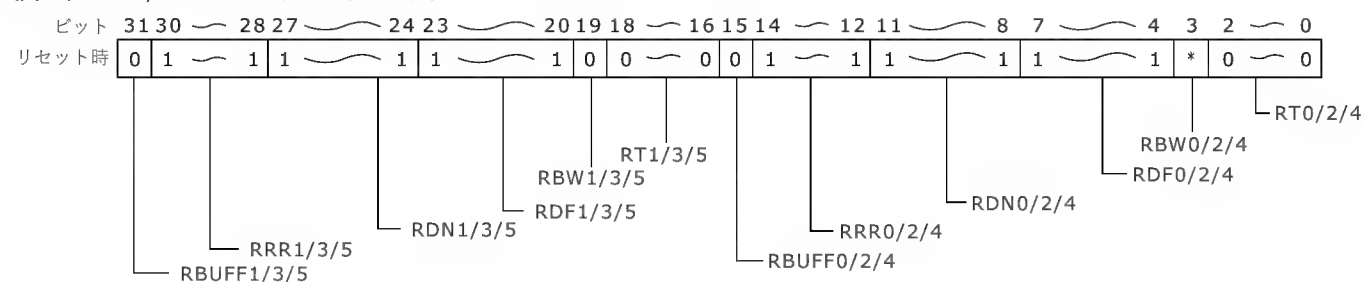
図11にCPUボードのメモリマップを示します．CPUはリセット直後，CS0の空間にアクセスしにいくので，ここにはROMが必要です．CPUボードではフラッシュメモリを接続しています．CS0のバス幅などの設定はCPUのBOOT_SEL信号ピンで行いますが，CPUボードのフラッシュメモリはROMソケットではなくフラットパッケージを実装し交換を想定していないので，CPUボードの仕様としてBOOT_SEL[2：0]は“000”で固定しています．

RAMはSDRAMバンク0にPC100，CL=3の仕様のSDRAMを接続しています．さらにCS2の空間には後述するポート機能レジスタをマッピングしています．

● メモリコンフィグレーションレジスタの設定

表5にメモリコンフィグレーションレジスタの設定を示します．基本的にこれらの設定は，フラッシュメモリ内にあるモニタプログラムの初期化部分で設定されるので，SDRAMにダウンロードして実行するユーザープログラムで初期化する必要は

〔図11〕CPUボードのメモリマップ

〔表5〕メモリコンフィグレーションレジスタの設定値

レジスタ名	アドレス	設定値
MDCNFG	0x48000000	0x1AA1
MDREFR	0x48000004	0x03C110C8→0x038190C8
MSC0	0x48000008	0x23F023F0
MSC1	0x4800000C	0x23312331
MSC2	0x48000010	初期値（未使用）
MECR	0x48000014	初期値（未使用）
SXCNFG	0x4800001C	初期値（未使用）
SXMRS	0x48000024	初期値（未使用）
MCMEM0	0x48000028	初期値（未使用）
MCMEM1	0x4800002C	初期値（未使用）
MCATT0	0x48000030	初期値（未使用）
MCATT1	0x48000034	初期値（未使用）
MCIO0	0x48000038	初期値（未使用）
MCIO1	0x4800003C	初期値（未使用）
MDMRS	0x48000040	0x1900

〔図12〕MDCNFG レジスタのフォーマット

〔図13〕MDREFR レジスタのフォーマット

〔図14〕MDMRS レジスタのフォーマット

〔図15〕MSC0/MSC1 レジスタのフォーマット

ありません．しかし，完成したプログラムをフラッシュメモリに書き込んで実行する場合は，これらのレジスタを初期化しないと，SDRAMや後述するポート機能が使えません．

MDCNFGはSDRAMのコンフィグレーションを行うレジスタです(**図12**)．SDRAMバンク0/1側と2/3側の設定が可能です．ここでは，各SDRAMバンクの有効/無効，バス幅，ロウ/カラムアドレス幅，バンク数，レイテンシなどの設定を行います．

評価ボードでは，64Mビットの SDRAMが実装されており，16ビット×2の32ビット幅，ロウアドレス12ビット，カラムアドレス8ビット，4バンクの構成です．CASレイテンシは3で設定されています．

MDREFRはSDRAMのリフレッシュ関連の設定を行うレジスタです(**図13**)．リフレッシュ間隔の設定，SDCLKの有効，クロックイネーブルの有効の設定などを行います．また，SDCLKを内部メモリ周波数と同じにするか1/2にするかの設定もここで行います．

表5中に“0x03C110C8→0x038190C8”とあるのは，最初，セルフリフレッシュでクロックのみ供給した後，セルフリフレッシュをディセーブルとし，クロックイネーブルを有効にするものですが，ほかのMDCNFGやMDMRSと合わせ，8回のダミーサイクルを入れたり，ウェイトを入れたりなど，正しくはSDRAMの初期化シーケンスに沿って行う必要があります．

MDMRSは，SDRAMのモードレジスタ設定用のレジスタです(**図14**)．MDCNFGでSDRAMバンクを有効にした後に設定します．MDCNFGで設定されたCLの設定値が使用されます．バースト長は4に固定されています．

MSC0はCS0/1の設定を行うレジスタです(**図15**)．評価ボードでは，CS0は32ビットバス幅でフラッシュメモリが実装されています．評価ボードの場合はノンバーストROM，フルウェイトに設定しています．これは，高速なフラッシュメモリを入手できず，やむなくアクセス速度の遅い，またはバーストアクセスに非対応のフラッシュメモリを実装することになっても，ROMの内容は変更しなくても動作するようにとの措置です．

このレジスタはユーザープログラムから書き換えることができるので，実機のフラッシュメモリの仕様に合わせて最適なパフォーマンスに変更することができます．現在出荷中のCPUボードの場合は，4バーストROMで，ウェイト13の設定が可能です．

CS1はボード上では未使用ですが，CS0と同様の設定をしています．

MSC1は，CS2/3の設定を行うレジスタです．評価ボードではCS2領域にポート機能があります．設定は32ビット幅SRAM設定で，ウェイトは3です．

● GPIO機能

表6に，CPUボードで使用しているGPIO機能を示します．PXA250にはGPIOが合計81本ありますが，表のようにほかのさまざまな周辺機能のI/Oピンと兼用になっているので，使用したい機能が同じピンに重ならないように，使い分ける必要があります．

たとえば，できるだけ多くのGPIO機能を使いたいという場合は，デバッガとの接続インターフェースとしてUSB接続を選択し，後述する方法でCPUボード上のRS-232-Cドライバをディセーブル状態にすれば，FFUARTおよびBTUARTで使われるI/OピンをすべてGPIOのI/Oピンとして使うことが可能です．

● ポート機能

CQ RISC評価キット/XScaleのCPUボードに実装されている7セグメントLEDやディップスイッチは，PXA250のGPIO機能を使って直接制御しているわけではありません．これらの制御はCS2の空間にメモリマップドI/Oで制御レジスタを実装しています．

表7にポート機能制御レジスタを示します．CS2の空間は

〔表6〕CPUボードで使用しているGPIO機能

GPIO	名　称	I/O	機　能
0	USB_ON	I	USB電源を検出したとき1，通常0
1	SW_BRKRQ	I	BRKRQスイッチが押されたとき0，通常1
4	RS232VALID	I	RS-232-Cドライバが有効なとき1，無効のとき0
15	nCS1	O	nCS1　PLDでバッファの制御をしている
33	nCS5	O	nCS5　PLDでバッファの制御をしている
34	FFRXD	I	FF UART RXD
35	FFCTS	I	FF UART CTS
36	FFDCD	I	FF UART DCD
37	FFDSR	I	FF UART DSR
38	FFRI	I	FF UART RI
39	FFTXD	O	FF UART TXD
40	FFDTR	O	FF UART DTR
41	FFRTS	O	FF UART RTS
42	BTRXD	I	BT UART RXD
43	BTTXD	O	BT UART TXD
44	BTCTS	I	BT UART CTS
45	BTRTS	O	BT UART RTS
46	ICP_RXD	I	IrDA RXD
47	ICP_TXD	I	IrDA TXD
52	nPCE1	O	nPCE1　PLDでバッファの制御をしている
53	nPCE2	O	nPCE2　PLDでバッファの制御をしている
78	nCS2	O	nCS2　ポートがマッピングされている
79	nCS3	O	nCS3　PLDでバッファの制御をしている
80	nCS4	O	nCS4　PLDでバッファの制御をしている

〔表7〕ポート機能制御レジスタ

アドレス	名　称
0x0A000000	コンフィグレーションポート
0x0A000010	7セグメントLEDポート
0x0A000018	ディップスイッチポート

〔表8〕7セグメントLEDの点灯制御レジスタ

ビット	名　称	機　能
0	LED0	0のライトで7セグメントLEDの"A"が点滅，1のライトで消灯
1	LED1	0のライトで7セグメントLEDの"B"が点滅，1のライトで消灯
2	LED2	0のライトで7セグメントLEDの"C"が点滅，1のライトで消灯
3	LED3	0のライトで7セグメントLEDの"D"が点滅，1のライトで消灯
4	LED4	0のライトで7セグメントLEDの"E"が点滅，1のライトで消灯
5	LED5	0のライトで7セグメントLEDの"F"が点滅，1のライトで消灯
6	LED6	0のライトで7セグメントLEDの"G"が点滅，1のライトで消灯
7	LED7	0のライトで7セグメントLEDの"DP"が点滅，1のライトで消灯
8～31	−	未使用

(a) ビット割り当て

(b) セグメントの配置

〔表9〕ディップスイッチ入力制御レジスタ

ビット	名　称	機　能
0	MONI_ON	1：モニタON，0：モニタOFF
1	SERIAL_USB	1：シリアルデバッグ時，0：USBデバッグ時
2	DIPSW3	DIPSW3　1：ON，0：OFF
3	DIPSW4	DIPSW4　1：ON，0：OFF
4	DIPSW5	DIPSW5　1：ON，0：OFF
5	DIPSW6	DIPSW6　1：ON，0：OFF
6	DIPSW7	DIPSW7　1：ON，0：OFF
7	DIPSW8	DIPSW8　1：ON，0：OFF
8～31	−	未使用

〔表10〕コンフィグレーションポートレジスタ

ビット	名　称	機　能
0/1	−	未使用
2	IRDA_FSEL	HDSL-3600のFIR_SELの設定
3	IRDA_MD0	HDSL-3600のMD0の設定
4	IRDA_MD1	HDSL-3600のMD1の設定
5～8	−	未使用
9	RS232ON	RS-232-Cドライバを有効とするとき1，無効とするとき0
10/11	−	未使用
12	LED1	0のライトでLED1(赤)が点灯，1のライトで消灯
13	LED2	0のライトでLED2(緑)が点灯，1のライトで消灯
14～31	−	未使用

0x08000000から0x0BFFFFFFまでの64Mバイトありますが，その真中の0x0A00000付近に配置しています．CPUボード上に実装しているCPLDでアドレスデコードを行っています．

● 7セグメントLEDとディップスイッチ入力

表8に7セグメントLEDの点灯制御レジスタを，**表9**にディップスイッチ入力制御レジスタを示します．

7セグメントLEDはビットごとに点灯を制御でき，ビットにゼロを書き込むとセグメントが点灯します．よって数値を表示したいときは，表示する数値に対応したセグメント点灯パターンの値を書き込む必要があります．

ディップスイッチはON状態で1，OFF状態で0が読み出されます．なお，ビット0と1はCPUボードのモニタROMで用途を規定しているので，ユーザーが自由に切り替えることはできません．もちろん，モニタROMを使わず，フラッシュメモリにユーザーの作成したプログラムをROM化して書き込んで使う場合は，8ビットすべてを自由に使うことができます．

● コンフィグレーションポート

もう一つ重要なレジスタに，コンフィグレーションポートがあります．**表10**に，コンフィグレーションポートのレジスタを示します．

このポートでは，CPUボード上のIrDAモジュールの機能や，RS-232-Cドライバの設定，赤と緑のLEDの点灯状態を制御することができます．

たとえば，シリアルポートJCOM1を使わないのであれば，コンフィグレーションポートのビット9を0にすることで，JCOM1に接続されているRS-232-Cドライバがディセーブル状態になります．この状態でFFUARTで使われているI/Oピンは完全にフリーになるので，各ピンをGPIOとして自由に使うことが可能になるわけです．

まとめ

以上，PXA25x/PXA26xアプリケーションプロセッサのハードウェアとCQ RISC評価キット/XScaleのCPUボードの仕様について解説しました．誌面の都合などで，今回は概要の紹介のみとなってしまいました．さらに詳細な仕様を知りたい方は，参考文献を参照ください．

次号では，CQ RISC評価キット/XScaleを使用したプログラム作成とデバッグについて解説します．

参考文献

1) *Intel (r) XScale Microarchitecture for the PXA255 Processor User's Manual*
2) *Intel (r) PXA255 Processor Developer's Manual*
3) *Intel (r) PXA255 Processor Design Guide*
4) *Intel (r) PXA26x Processor Family Developer's Manual*
5) *Intel (r) PXA26x Processor Family Design Guide*

ほさか・かずひろ　(株)ソフィアシステムズ

CQ RISC評価キット

SH-4PCI with Linux活用研究4

Microwindowsを使った組み込み向けGUIプログラムの作成事例(応用編)

酒匂信尋

基礎編である前回は，Microwindowsを使ったもっとも基本的な組み込み機器向けGUIプログラムの作成事例を紹介した．応用編となる今回は，ディップスイッチやLED点灯制御が可能なPCIボードを実装し，デバイスドライバと組み合わせてより実用的な組み込み機器向けGUIプログラムの作成事例を紹介する．（編集部）

はじめに

前回の基礎編で，Microwindows(Nano-X)で動作するGUIプログラムを作成しました．今回は，ハードウェアと連携したGUIプログラムに拡張してみることにします．

ハードウェアへの入出力は割り込み待ちなどがあるので，Microwindowsのイベントループ内で直接ハードウェアへアクセスするのは適切ではありません．一般的に，このような場合は複数のプログラムを並列に動作させることで実現します．

従来の組み込み用OSであれば，別タスクを作成し，そこでハードウェアの入出力を行うことになると思います．Linuxのような UNIX系のOSの場合，一昔前のBSD系UNIXであれば，`fork()`で別プロセスを作成し，ソケット通信でプロセス間の情報を授受し，情報量が多い場合は(苦労して？)プロセス間の共有メモリを作成していたと思います．

このように処理の単位をプロセス単位にすることで，一つのプロセスの異常が他のプロセスに与える影響を抑えることができます．これはシステムの信頼性向上の意味で有効な構造ではありますが，資源の有効利用やCPUへの負荷を考えると，`fork()`などを用いた複数プロセスによる並列処理は，小規模な組み込みシステムには適切でないように思えます．

それと比較してスレッドによる並列処理は，同一のメモリ空間を使うため，とくに意識せずにスレッド間の共有メモリを実現できます．ただし，スレッド間でお互いの情報を壊してしまう可能性もあります．このように，メモリ保護が働かないという意味では従来の組み込み用OSでのプログラミングと似ているともいえるので，組み込み機器の技術者にとっては，従来のUNIXの作法の並列処理プログラミングより，スレッドを使った構造のほうが違和感がないのではないかと思います．

〔図1〕カーネルのLoadable module supportの設定

Loadable module support

y	m	n		
◆ y	-	n	Enable loadable module support	Help
y	-	◆ n	Set version information on all module symbols	Help
y	-	◆ n	Kernel module loader	Help

Main Menu　Next　Prev

そこで今回は，簡単なデバイスドライバを作成し，POSIXスレッドを用いてデバイスをコントロールするGUIプログラムを作成してみることにします．

1 CQ RISC評価キット/SH-4PCI with Linuxでの開発環境整備

● コンソールの増設やPATHの設定など

すでに前回の基礎編で説明した設定なので，詳細は省略します．

● 画面サイズと色数

Nano-Xサーバでの画面解像度や表示色数は，Linuxカーネルのフレームバッファで決定されます．カーネルのソース〔`linux/drivers/video/chipsfb.c`の`init_chips()`〕を見ると，評価キットでは800×600ドットの8ビットカラーとなっています．ここで，もし実際の画面サイズが600×600ドットで表示されている場合は，評価ボードに搭載されているビデオチップの初期化ルーチンにバグがあります．次のURLでパッチが公開されているので，パッチをあててカーネルを再構築してください．

`http://www.cqpub.co.jp/eda/CqREEK/sh4pci/linux.htm`

● モジュール対応のカーネルを再構築

今回はドライバをモジュールとして作成しますが，評価キットのCD-ROMからそのままインストールした状態では，カーネルがモジュール対応になっていません．そこで，カーネルの再構築を行います．

`xconfig`で`loadable module support`を選択し，**図1**のようにイネーブルに設定します．具体的な手順は，下記のようになります．

```
# /cd/opt/lineo-BDK/KMC-BDK/sys/linux
# make xconfig
# make dep
# make
```

新しく作成した`vmlinux`を，ftpなどで評価キットに転送して再起動します．そして`lsmod`を実行してみます．

〔リスト1〕修正したmake.common

```
# Generated automatically from Makefile.common.in by configure.
# Common code included in every Makefile

DESTDIR          =

exec_prefix      = ${prefix}
insmod_static    = no
mandir           = ${prefix}/man
prefix           = /opt/sh4src/modutils-2.4.22/build/target
sbindir          = ${exec_prefix}/sbin

AR               = sh4-linux-ar
ARCH             = sh
CC               = sh4-linux-gcc
CFLAGS           = -O2 -Wall
# HOSTCC and HOSTCFLAGS are provided for compatibility.
HOSTCC           = sh4-linux-gcc
HOSTCFLAGS       = -O2 -Wall
BUILDCC          = gcc
BUILDCFLAGS      = $(HOSTCFLAGS)
INSTALL          = install
LDFLAGS          =
LIBS             =
MKDIR            = mkdir -p
TAINT_URL        = http://www.tux.org/lkml/¥#export-tainted
PARSERCFLAGS     = -Wno-uninitialized
RANLIB           = sh4-linux-ranlib
STRIP            =
USE_SYSCALL      = n
```

```
# lsmod
Module                  Size  Used by
lsmod: QM_MODULES: Function not implemented
```

上のようなエラーが表示された場合は，モジュールが正しく組み込まれていません．カーネル再構築の手順などを確認してください．

● モジュール操作用ツールの移植

lsmodは評価キットにはじめからインストールされているのですが，モジュールのインストール(insmod)やモジュールの消去(rmmod)などについてはインストールされていないので，移植を行います．ソースについては，次のサイトなどを参照してください．

ftp://ftp.kernel.org/pub/linux/utils/kernel/modutils/V2.4

手順としては，次のようになります．

```
# cd /opt/sh4src
# tar xzvf ??/modutils-2.4.22.tar.gz
# cd modutils-2.4.22
# mkdir build
# cd build
# ../configure  --target=sh4-linux
      --prefix=/opt/sh4src/modutils-2.4.22/build/target
# make
# make install
```

評価キットのHDDのbuild/targetの下に，作成されたツール類を転送します．

なお，これらの作業の中，作成されたMakefileなどに不具合があったので，makeを実行する前に次の修正を行います．

▶ make.commonの修正

クロスツールの指定とインストール時のストリップ指定をはずします．修正した結果を**リスト1**に示します．

▶ PATH_MAX，ST-TYPEのアンデファイン対策

util/logger.c，utils/meta-expand.c，genksyms/lex.l，depmod/depmod.cなどの各ソースファイルでエラーが出ます．各ファイルでインクルードされているlimits.hのパスを次のように修正します．

include <limits.h> → include <linux/limits.h>

またinsmod.cのST_TYPEの部分でもエラーが出ます．360行目のELFW(ST_TYPE)の部分をELF32_ST_TYPEに修正します．パッチファイルを**リスト2**に示します．

2 デバイスドライバの作成

● 単純なI/Oカードのドライバは簡単

デバイスドライバというと，非常に難しいと思っている方が多いと思います．実際，通信関連などのアプリケーションインターフェースの間にプロトコルスタックが入る場合は，作法がきっちりと決められますから，それに合わせて作る必要があります．

デバイスドライバについてはいくつか解説書も市販されていますが，どういうわけかバージョン2.4.xがリリースされた頃に2.2.x対応の解説書が発売というようなケースが多いようです(バージョンアップに対応してドライバを作成して原稿を執筆し，印刷して発売……という頃には，もうカーネルのバージョンが上がったということか)．このような場合は，類似ソースを参考に作成することになります．

今回テストで使用するPCIボードのような，独自仕様かつ単純なI/Oポートのリード/ライトのような場合は，何らかの規格

〔リスト2〕insmod.cの修正部分

```
--- insmod.c.org  Thu Mar  6 00:04:16 2003
+++ insmod.c Thu Mar  6 00:04:45 2003
@@ -357,7 +357,7 @@

     for (p = specials; *p; ++p)
         if ((sym = obj_find_symbol(f, *p)) != NULL)
-            sym->info = ELFW(ST_INFO) (STB_LOCAL, ELFW(ST_TYPE) (sym->info));
+            sym->info = ELFW(ST_INFO) (STB_LOCAL, ELF32_ST_TYPE(sym->info));
 }

 static void print_load_map(struct obj_file *f)
```

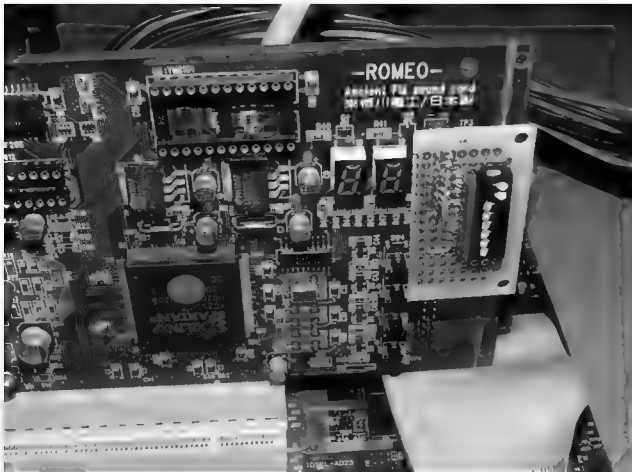
〔写真1〕テスト用DIOボードの入出力部のようす

〔表1〕テスト用DIOボードの仕様

ベンダID	6809h
デバイスID	8000h
ベースアドレスレジスタ0	I/O空間を16バイト要求
割り込み	INTA#使用

(a) コンフィグレーションレジスタ仕様

オフセット	アクセスサイズ（ビット）	機　能
+00	16	7セグメントLED点灯出力
+02	16	ディップスイッチ入力 実際には下位8ビットのみ有効
+04	16	割り込みステータスレジスタ（下位4ビット） '1'で割り込み発生 '1'を書き込むと割り込みクリア
+06	16	割り込みマスクレジスタ（下位4ビット） '1'で割り込み出力許可 '0'で割り込み出力禁止
+08以降	–	予約

(b) I/O仕様

やインターフェース，APIなどに合わせる必要はなく何のしがらみもないので，簡単に作成することができます．ここでは本誌2003年3月号の「PCIデバイス対応デバイスドライバの作成法」〔参考文献2)〕を参考に作成します．

● テスト用DIOボードの仕様

今回使用したテスト用のPCIボードは，8ビットのディップスイッチ入力と，2桁の7セグメントLED，4個のプッシュスイッチによる割り込み発生機能を実装したディジタルI/O（DIO）ボードです（**写真1**）．DIOボードの仕様を**表1**に示します．

16ビットの入力ポートを読み出すと下位8ビットにディップスイッチの状態が，16ビットの出力ポートに書き込むと7セグメントLEDが点灯します．7セグメントLEDは，ドットを含めた八つのセグメントがそのまま各ビットに対応しています．2桁の7セグメントLEDがあるので，出力ポートは16ビット全ビットが動作します．

割り込みを発生させるプッシュボタンは，それぞれ独立して割り込みマスクビットやステータスビットをもっています．プッシュボタンを押すと，対応したステータスビットが'1'になります．クリアする場合には，該当ビットに'1'を立てた値を書き込みます．また，割り込みマスクビットが割り込み許可状態に設定されていると，PCIバス上に割り込みを発生させます．

● リソースの設定

評価キットはPC/AT互換機のようなPCI BIOSをもっているわけではないので，PCIスロットにPCIボードを実装しても，そのボードに対してアドレスや割り込みが割り当てられません．PCIデバイスはこれらリソースが割り当てられないと，まったく使用することができません．

リソースの割り当ては`linux/arch/ch/setup_kzp01.c`で行います．処理の内容は，PCIのコンフィグレーションレジスタをチェックして該当するベンダID，デバイスIDをもっているデバイスを探します．見つかったら，I/Oアドレスを割り当て，コマンドレジスタのI/O空間イネーブルビットをイネーブルに設定し，さらに割り込みを設定します．割り込みはIRQ10を使用するようにしています．具体的な追加プログラムを**リスト3**に示します．

修正が終わったら，環境整備の項と同様の手順でカーネルの再構築を行います．これが終了すれば，しっかりしたPCI BIOSをもっているPC/AT互換機と同等のスタートラインについたことになります．

● ドライバの作成

ドライバは，次の関数で構成します．

```
init_module()
cleanup_module()
open()
release()
read()
write()
ioctl()
```

▶ init_module()

カーネルにドライバの関数を登録します．登録には`file_operations`（**リスト4**）という構造体を用います．カーネルに大きなバージョンアップがあるとフィールドの追加があります．バージョン2.4.xの場合は17個もの関数が登録できるようになっています．今回は自前のアプリケーションから呼び出すだけですから，必要最低限の関数（`open`，`release`，`read`，`write`，`ioctl`）だけを書くことにします．

実際の登録はデバイスのメジャー番号，デバイス名称とドライバの関数をセットした`file_operations`の構造体を引き数にして`register_chedev()`を用います．

▶ cleanup_module()

`init_module()`は`insmod`時に呼び出されますが，この関数は`rmmod`でドライバを削除するときに，呼び出されます．`unregister_chrdev()`で登録を削除します．

〔リスト3〕リソースの設定

```
--- setup_kzp01.c.org  Wed Mar 12 14:57:46 2003
+++ setup_kzp01.c Wed Mar 12 15:21:02 2003
@@ -179,6 +179,29 @@
      outb(0x03,0x3c2);                            /* ram & i/o disable */
 #endif /* } */

+
+/*************** CQ test dio unit ***hajimari*****************************/
+{
+    unsigned int  vendorid = 0x80006809;
+    unsigned char bus_no = 1;
+    unsigned char dev_no;
+    unsigned int tmp;
+   for( dev_no = 0; dev_no < 16; dev_no++) {
+       tmp = pci_cfgrd(bus_no,dev_no,0,PCI_VENDOR_ID);
+       if( tmp == vendorid) {
+        pci_cfgwr(bus_no,dev_no,0,PCI_BASE_ADDRESS_0,0x2000);
+        pci_cfgwr(bus_no,dev_no,0,PCI_COMMAND_MASTER,0x02000001);
+        pci_cfgwr(bus_no,dev_no,0,PCI_INTERRUPT_LINE, 0x0a);
+        val = pci_cfgrd(bus_no,0x02,0,0x48);
+        pci_cfgwr(bus_no,0x02,0,0x48, val | 0x00000003 << 4*(dev_no - 5));
+        val = inb(0x4d1);
+        outb(val | 0x04,0x4d1);
+        break;
+       }
+   }
+}
+/*************** CQ test dio unit ***owari********************************/
+
+
 /*
  * pci device setup owari !!
 */
```

〔リスト4〕file_operations構造体

```
/*
 * NOTE:
 * read, write, poll, fsync, readv, writev can be called
 *   without the big kernel lock held in all filesystems.
 */
struct file_operations {
    struct module *owner;
    loff_t (*llseek) (struct file *, loff_t, int);
    ssize_t (*read) (struct file *, char *, size_t, loff_t *);
    ssize_t (*write) (struct file *, const char *, size_t, loff_t *);
    int (*readdir) (struct file *, void *, filldir_t);
    unsigned int (*poll) (struct file *, struct poll_table_struct *);
    int (*ioctl) (struct inode *, struct file *, unsigned int, unsigned long);
    int (*mmap) (struct file *, struct vm_area_struct *);
    int (*open) (struct inode *, struct file *);
    int (*flush) (struct file *);
    int (*release) (struct inode *, struct file *);
    int (*fsync) (struct file *, struct dentry *, int datasync);
    int (*fasync) (int, struct file *, int);
    int (*lock) (struct file *, int, struct file_lock *);
    ssize_t (*readv) (struct file *, const struct iovec *, unsigned long, loff_t *);
    ssize_t (*writev) (struct file *, const struct iovec *, unsigned long, loff_t *);
    ssize_t (*sendpage) (struct file *, struct page *, int, size_t, loff_t *, int);
    unsigned long (*get_unmapped_area)(struct file *, unsigned long, unsigned long, unsigned long, unsigned long);
};
```

▶ open()

割り込みハンドラの登録を行います.

▶ release()

open()で登録した割り込みハンドラを削除します.

▶ read()/write()

ハードは単純なDIOボードですから，ポートへの入出力を行います.

▶ ioctl()

read時，割り込み待ちの有無や割り込み解除などのread/writeでは行わない，動作状況の変更を行います.

▶その他

open()，release()でMOD_INC_USE_COUNT，MOD_DEC_USE_COUNTというマクロを呼んでいます. これは利用回数のことで，カーネルのモジュール管理で使用します. 記述がないと動かないというわけではありませんが，ドライバ作成の作法と考えてください.

作成したプログラムを**リスト5**に，コンパイル方法を**リスト6**に示します.

● デバイスファイルの作成

作成したドライバは，デバイス名“dio”，メジャー番号を206としました. 作成手順は次のようになります.

```
# cd /dev
# mknod dio c 206 0
```

〔リスト5〕ドライバのソース

```
/*
 * dio_drvr.c
*/

#ifndef __KERNEL__
  #define __KERNEL__
#endif

#ifndef MODULE
  #define MODULE
#endif

#define __NO_VERSION__

#define EXTERN_INLINE extern __inline__

#include <linux/module.h>
#include <linux/version.h>

#include <asm/io.h>
#include <asm/uaccess.h>
#include <asm/segment.h>

char kernel_version[] = UTS_RELEASE;

#include <linux/sched.h>
#include <linux/proc_fs.h>
#include <linux/pci.h>
#include <linux/fs.h>
#include <linux/vmalloc.h>
#include <linux/compatmac.h>


#define DIO_DRVR_INT_ENABLE 1
#define DIO_DRVR_INT_DISABLE     2
#define DIO_DRVR_SOFT_INT   3

unsigned char bus, func;
struct pci_dev *dev;
unsigned short vendor = 0x6809, id = 0x8000;
unsigned int vendorid = 0x80006809;
#define DIO_DRVR_MAJOR  206
char irq_no ;

unsigned long mem_start;
unsigned long io_start;
int int_count;

DECLARE_WAIT_QUEUE_HEAD(dio_drvr_q);
int int_enable;

static unsigned long
pci_config_dread(unsigned char bus_no,
          unsigned char dev_no,
          unsigned char adr)
{
     unsigned long x;
     x = 0x80000000 | ((unsigned long)bus_no<<16)
                    | ((unsigned long)dev_no<<11)
                    |  adr;
    *(unsigned long *)0xfe2001c0 = x;
    return *(unsigned long *)0xfe200220;
}
/*
static void
pci_config_dwrite(unsigned char bus_no,
          unsigned char dev_no,
          unsigned char adr,
      unsigned long data)
{
     unsigned long x;
     x = 0x80000000 | ((unsigned long)bus_no<<16)
                    | ((unsigned long)dev_no<<11)
                    |  adr;
    *(unsigned long *)0xfe2001c0 = x;
    *(unsigned long *)0xfe200220 = data;
}
*/
void dio_drvr_interrupt(int irq, void *dev_id, struct pt_regs *regs)
{
  printk(KERN_DEBUG "dio interrupt %x ¥n",inw(io_start + 4));
  if (!(inw(io_start + 4)))
    return;
  outw(inw(io_start + 4) ,io_start + 4);
  wake_up_interruptible(&dio_drvr_q);
  int_count++;
}

int dio_drvr_open(struct inode *inode, struct file *file)
{
  unsigned int tmp = 0;
  unsigned int bus_no,dev_no;

 bus_no = 1;
 for( dev_no = 0; dev_no < 16 ; dev_no++) {
      tmp = pci_config_dread(bus_no,dev_no,PCI_VENDOR_ID);
      if (tmp == vendorid) {
    printk(KERN_DEBUG "dio_drvr : device found. bus_no = 0x%x, dev_no = 0x%x¥n",
              bus_no, dev_no);
    break;
      }
  }
  if (tmp != vendorid) {
      printk(KERN_DEBUG "dio_drvr : device not found!!¥n");
      return (-ENODEV);
  }

  io_start = pci_config_dread(bus_no,dev_no,PCI_BASE_ADDRESS_0) & 0xfffffffc;
  irq_no   = pci_config_dread(bus_no,dev_no,PCI_INTERRUPT_LINE) & 0x0f;
  printk(KERN_DEBUG "dio_drvr :  io = 0x%04x, irq_no = 0x%02x¥n",
                        (int)io_start, irq_no);

  if (request_irq(irq_no, dio_drvr_interrupt, SA_INTERRUPT,
        "dio", NULL) != 0) {
    printk(KERN_DEBUG "dio_drvr : request_irq() error¥n");
    return (-ENODEV);
  } else {
    printk(KERN_DEBUG "dio_drvr : request_irq() succeeded¥n");
  }

  printk(KERN_DEBUG "dio_drvr : device found. vendor = 0x%x, id = 0x%x¥n",
     vendorid & 0x0000ffff, vendorid >> 16);
```

〔リスト5〕ドライバのソース（つづき）

```
  MOD_INC_USE_COUNT;
  return (0);
}

int dio_drvr_release(struct inode *inode, struct file *file)
{
  printk(KERN_DEBUG "dio_drvr : release¥n");
  outw(inw(io_start + 6) & ~0x0f, io_start + 6);
  free_irq(irq_no, NULL);
  MOD_DEC_USE_COUNT;
  return (0);
}

ssize_t dio_drvr_read(struct file *file, char *buf, size_t len, loff_t *f_pos)
{
  unsigned short val;

  if (int_enable)
    interruptible_sleep_on(&dio_drvr_q);
  val = inw(io_start + 2);
  *buf = (char)val;
  *f_pos++;
  return (1);
}

ssize_t dio_drvr_write(struct file *file, const char *buf, size_t len, loff_t *f_pos)
{
  outw(*(unsigned short *)buf,io_start);
  *f_pos++;
  return (1);
}

/* ioctl
 */
int dio_drvr_ioctl(struct inode *inode, struct file *file,
         unsigned int cmd, unsigned long arg)
{

  switch (cmd) {
  case DIO_DRVR_INT_ENABLE:
    outw(inw(io_start + 6) | 0x0f, io_start + 6);
    int_enable = 1;
    break;
  case DIO_DRVR_INT_DISABLE:
    outw(inw(io_start + 6) & ~0x0f, io_start + 6);
    int_enable = 0;
    break;
  case DIO_DRVR_SOFT_INT:
    wake_up_interruptible(&dio_drvr_q);
    break;
  default:
    return -EINVAL;
  }
  return 0;
}


struct file_operations dio_drvr_fops = {
  ioctl: dio_drvr_ioctl,
  read:      dio_drvr_read,
  write: dio_drvr_write,
  open:      dio_drvr_open,
  release:   dio_drvr_release,
};

int init_module(void)
{
  int result;

  result = register_chrdev(DIO_DRVR_MAJOR, "dio", &dio_drvr_fops);
  if (result < 0) {
    printk(KERN_DEBUG "dio_drvr.o : unable to get major number.%x¥n",result);
    return 0;
  }
  printk(KERN_DEBUG "dio_drvr : module loaded %x¥n",result);
  return 0;
}

void cleanup_module(void)
{
  unregister_chrdev(DIO_DRVR_MAJOR, "dio");
  return;
}
```

〔リスト6〕ドライバのコンパイル方法

```
#! /bin/sh
sh4-linux-gcc -D__KERNEL__ -Wall -Wstrict-prototypes -O2 -fomit-frame-pointer -fno-strict-aliasing -ml -m4-nofpu -pipe -DMODULE
-c -o dio_drvr.o dio_drvr.c
```

● デバイスドライバの機能確認

まず，デバイスドライバが正しく動作しているか確認します．機能の確認用プログラムも参考文献2)を参考に作成します．機能的にも同等のものなので，解説は省略します．作成したテストプログラムを**リスト7**に示します．

テストプログラムのコンパイルは，次のようになります．

```
# sh4-linux-gcc -o dtest dtest.c
```

動作確認の方法は次のとおりです．

```
# insmod dio_drvr.o
# ./dtest
> w 0   ← LEDが点灯
> r
c0 ← 読み取ったディップスイッチの内容
> a
c0 ← 割り込みボタンを押したときに読み取ったディップ
     スイッチの内容
```

評価キットには三つのPCIスロットがありますが，3スロット目(CN14)では割り込みが取れませんでした．CN14はオンボードのLANコントローラとINTA #が共用になっているのが問題のようです．これについても参考文献2)に解説があるので，参照してください．

〔リスト7〕ドライバテストプログラム

```
#include <fcntl.h>
#include <stdio.h>
#include <unistd.h>
#include <linux/fs.h>

int arg;

#define  DIO_DRVR_INT_ENABLE 1
#define  DIO_DRVR_INT_DISABLE 2

void do_command(int fd, char *cmd, unsigned int data)
{
  unsigned char buf[2];
  int i, len, arg;


if (cmd[0] == 'r') {
    read(fd, buf, 1);
    printf(" %02x", buf[0]);
} else if (cmd[0] == 'w') {
    write(fd, &data, 2);
} else if (cmd[0] == 'a') {
    ioctl(fd, DIO_DRVR_INT_ENABLE);
    read(fd, buf, 1);
    printf(" %02x", buf[0]);
    ioctl(fd, DIO_DRVR_INT_DISABLE);
  }
  printf("¥n");
}

int main(int argc, char **argv)
{
  FILE *input_stream = NULL;
  char cmd[10], buf[256];
  unsigned int data;
  int fd;

  if ((fd = open("/dev/dio", O_RDWR)) == -1) {
    fprintf(stderr, "Can't open /dev/dio¥n");
    exit(2);
  }

  while (printf("> "), fgets(buf, sizeof buf, stdin) != NULL) {
    cmd[0] = data = 0;
    sscanf(buf, "%s %x ", cmd, &data);
    do_command(fd, cmd, data);
  }

  close(fd);
}
```

3 ハードウェアと連携した GUIプログラミング

GUIプログラムのベースには前回のプログラムを流用し，ハードウェアと連携動作する部分を追加します．

● 追加する機能

DIOボードからの割り込みが発生したときに，データを取り込んで表示します．前回は用途未定だった二つのボタンに，ボタンをクリックしたときに入力ポートからデータを取り込むINPUTボタンと，出力ポートにデータを出力するOUTPUTボタンを作成します．

● スレッドの作成

スレッドは次の手順で作成します．

```
pthread_attr_init()
pthread_attr_setinheritsched()
pthread_attr_setschedpolicy()
pthread_create()
```

四つのシステムコールを使っていますが，実際はアトリビュート(スレッドの属性)関連の三つは現状では設定しなくても動きます．しかし近い将来の対応を考慮して，Linuxのmanで表示されるドキュメントに準じ，アトリビュートを設定したスレッドの作成を行います．

▶ pthread_attr_init()

デフォルト属性でスレッド属性の初期化を行います．

▶ pthread_attr_setinheritsched()

スレッド生成時のスケジューリングポリシーとパラメータを使用するかの設定です．

▶ pthread_attr_setschedpolicy()

スレッドが使用するスケジューリングポリシーを指定します．`SCHED_FIFO`はリアルタイム/ファーストインファーストアウト，`SCHED_RR`はリアルタイム/ラウンドロビン，`SCHED_OTHER`はデフォルトです．

一応，プログラム上はSCHED_RRを指定していますが，エラーにはなりません，きっと近い将来サポートされるのでしょう．

▶ pthread_create()

スレッドの作成を行います．attrのアトリビュートについては，上記三つの属性設定を省略した場合はNULLを指定します．あとはスレッドのアドレスと共有メモリのアドレスを指定します．

● スレッドの機能(main_dio)

デバイスドライバをopen()して，ioctl()で割り込みをイネーブル，その後read()で割り込み待ちとなります．割り込みが入ると入力ポートを読み込んで，共有メモリ上に取り込みデータと取り込みデータが更新されたことを記録します．

● GUIプログラムの修正

main()でスレッドの作成部分の追加のほかに，前回のプログラムでは用途未定となっていた二つのボタンに処理を追加します．一つ目はINPUTボタン，もう一つはOUTPUTボタンとします．

▶ INPUTボタンの機能(job_03_button)

ソフト割り込みを発生し，入力ポートの入力待ちでウェイトしているdio_main()にデータの取り込みを行わせます．

▶ OUTPUTボタンの機能(job_04_button)

dio_main()で取り込んだデータを出力ポートにwrite()で出力します．

ここで注意点がいくつかあります．今回の作成したデバイスドライバのwrite()はウェイトすることはないので，直接write()していますが，read()のように割り込み待ちなどでウェイトする可能性がある場合は，dio_main()のようにスレッドを作り，そちらに依頼するようにします．そうでないと，write()が完了するまでマウスやキーボードからのイベントが取れなくなります．

▶イベント待ちの変更

s_start()でのイベント待ちはGrGetNextEvent()を使用していましたが，このままだとマウスやキーボードの操作がないと，永久にイベント待ちとなってしまいます．そこで，タイマ指定のできる関数GrGetNextEventTimeout()に変更します．これで一定周期でイベント待ちから抜けてきます．抜けたタイミングでdio_mainスレッドの更新情報をチェックして，更新されていれば，入力情報を再表示します．

● 前回作成プログラムの問題点修正

前回作成したGUIプログラムは，再描画イベント発生時にボタンのウィンドウしか対応していないため，タイトルメッセージがウィンドウを重ねた後に消えてしまう現象がありました．

〔リスト9〕スレッドライブラリをリンクするMakefile

```
##############################################################################
# Microwindows template Makefile
# Copyright (c) 2000 Martin Jolicoeur, Greg Haerr
##############################################################################

include $(CONFIG)

######################## Additional Flags section ############################

# Directories list for header files
INCLUDEDIRS +=
# Defines for preprocessor
DEFINES += -DMWIN

# Compilation flags for C files OTHER than include directories
CFLAGS +=
# Preprocessor flags OTHER than defines
CPPFLAGS +=
# Linking flags
LDFLAGS +=

############################# targets section ################################

ifeq ($(NWIDGET), Y)
ifeq ($(NANOXDEMO), Y)

# If you want to create a library with the objects files, define the name here
LIBNAME =

# List of objects to compile
OBJS = cqsample.o

all: default $(TOP)/bin/cqsample

endif
endif

######################### Makefile.rules section #############################

include $(TOP)/Makefile.rules

######################## Tools targets section ###############################

$(TOP)/bin/cqsample: $(OBJS) $(NANOXCLIENTLIBS) $(TOP)/config
	$(CC) $(CFLAGS) $(LDFLAGS) $(OBJS) -o $@ $(NANOXCLIENTLIBS) -lpthread
```

今回はこの対応として，handle_exposure_event()にメインウィンドウと表示ウィンドウの再描画プログラムの追加を行います．

以上をまとめて，GUIプログラムを**リスト8**(稿末)に示します．

4 コンパイルと実行

● Makefileの修正

スレッドを使用したので，Makefileにスレッドライブラリのリンク(-lpthread)を追加します．修正したMakefileを**リスト9**(前頁)に示します．

● コンパイル

コンパイルは次のようにします．

```
# cd /opt/lineo-BDK/KMC-BDK/sys/microwin
# cd demos/cqsample
# make
```

以上の操作で，microwin/src/binにcqsampleが作成されます．

● 実行

実行手順は，次のようになります．

```
# insmod dio_drvr.o
# nano-X & sleep 1;  nxterm & nanown &
                                    sleep 10000
# ./cqsample &  ← xtermからの入力になる
```

実行時のようすを**図2**に示します．

まとめ

GUIからI/Oデバイスをコントロールするプログラムを，スレッドを使用して作成してみました．少し乱暴ですが，一つのプロセスを従来のITRONに代表されるような組み込み用OS，スレッドをタスクと置き換えると，スレッドによるプログラミングはUNIX系のプログラミングになじみがない方でも，かなりわかりやすいのではないかと思います．ITRONでcre_tsk()でタスクを作るのも，pthread_create()でスレッドを作るのもたいした違いはありません．難しいことはない！ということを感じていただければ幸いです．

〔図2〕ハードウェアと連携したGUIプログラムの動作画面

参考文献

1) 岸 哲夫，「SHプロセッサ用Linuxの概略と組み込み用GUIの解説」，Interface増刊『Embedded UNIX』Vol.2，CQ出版(株)
2) 竹内達也，田中 賢，「PCIデバイス対応デバイスドライバの作成法」，『Interface』，2003年3月，CQ出版(株)
3) 中野晃一，「UNIXとして設計されたRTOS LynxOS/VisualLynuxによるアプリケーション開発」，Interface 増刊『Embdded UNIX』Vol.2，CQ出版(株)
4)『LINUXデバイスドライバ』，(株)オライリー・ジャパン
5)『POSIXスレッドプログラミング』，アジソン・ウェスレイ・パブリッシャーズ・ジャパン(株)

さかわ・のぶひろ

〔リスト8〕GUIプログラムのソース

```
/*****************************************************************/
/*                                                               */
/*  Cq SH4/PCI Sample Program                                    */
/*                                                               */
/*****************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>
#include <time.h>
#include <ctype.h>
#include <errno.h>
#include <sys/time.h>
#include <pthread.h>
#include <sched.h>
#include <linux/fs.h>

#define MWINCLUDECOLORS
#include <nano-X.h>
#include "cqsample.h"
pthread_attr_t t_attr;
pthread_t      tid1;
#define INHERIT_SCHED 1
#define  DIO_DRVR_INT_ENABLE  1
#define  DIO_DRVR_INT_DISABLE 2
#define  DIO_DRVR_SOFT_INT    3

void draw_text(cstate *state)
{
    char buf[32];
    GrFillRect(state->window_02, state->win_02_gcb, 0, 0,
            WINDOW_02_WIDTH, WINDOW_02_HEIGHT);
    GrText(state->window_02, state->win_02_gcf, TEXT_X_POSITION,
            TEXT_Y_POSITION, "Event Counter", 13, 0);
    GrText(state->window_02, state->win_02_gcf, TEXT_X_POSITION,
            TEXT2_Y_POSITION, "Input Data", 10, 0);
    GrFillRect(state->window_01, state->win_01_gcb, 0, 0,
            WINDOW_01_WIDTH, WINDOW_01_HEIGHT);
    sprintf(buf, "%d", state->ev_cnt);
```

(a) cqsample.c

〔リスト8〕GUIプログラムのソース(つづき)

```
    GrText(state->window_01, state->win_01_gcf, TEXT_X_POSITION,
              TEXT_Y_POSITION, buf, strlen(buf), 0);
    sprintf(buf, "%d", state->input_data);
    GrText(state->window_01, state->win_01_gcf, TEXT_X_POSITION,
              TEXT2_Y_POSITION, buf, strlen(buf), 0);
}

void draw_01_button(cstate *state) /** start Button **/
{
    GrFillRect(state->button_01, state->buttongcb, 0, 0,
              BUTTON_01_WIDTH, BUTTON_01_HEIGHT);
    GrText(state->button_01, state->buttongcf, TEXT_X_POSITION,
              TEXT_Y_POSITION, "Satrt)", 5, 0);
}

void draw_02_button(cstate *state) /** stop Button **/
{
    if(!state->running_buttons_mapped) return;
    GrFillRect(state->button_02, state->buttongcb, 0, 0,
              BUTTON_02_WIDTH, BUTTON_02_HEIGHT);
    GrText(state->button_02, state->buttongcf, TEXT_X_POSITION,
              TEXT_Y_POSITION, "Stop", 4, 0);
}

void draw_03_button(cstate *state) /* input mode */
{
    if(!state->running_buttons_mapped) return;
    GrFillRect(state->button_03, state->buttongcb, 0, 0,
              BUTTON_03_WIDTH, BUTTON_03_HEIGHT);
    GrText(state->button_03, state->buttongcf, TEXT_X_POSITION,
              TEXT_Y_POSITION, "input", 5, 0);
}

void draw_04_button(cstate *state) /** LED output **/
{
    if(!state->running_buttons_mapped) return;
    GrFillRect(state->button_04, state->buttongcb, 0, 0,
              BUTTON_04_WIDTH, BUTTON_04_HEIGHT);
    GrText(state->button_04, state->buttongcf, TEXT_X_POSITION,
              TEXT_Y_POSITION, "output", 6, 0);
}

void job_02_button(cstate *state) /* stop */
{
    state->state = STATE_STOP;
    start_init(state);
}

void job_03_button(cstate *state) /* input */
{
    fprintf(stderr, "job 03 ¥n");
    ioctl(state->fd,DIO_DRVR_SOFT_INT);
}

void job_04_button(cstate *state) /* output */
{
    fprintf(stderr, "job 04 ¥n");
    write(state->fd,&state->input_data,2);
}

void handle_exposure_event(cstate *state)
{
    GR_EVENT_EXPOSURE *event = &state->event.exposure;
    if(event->wid == state->main_window) {
        GrText(state->main_window, state->maingcf,40,35,
        "CQ SH-4/PCI nano-X Sample program ", 34, 0);
        return;
    }
    if(event->wid == state->window_01) {
        draw_text(state);
        return;
    }
    if(event->wid == state->window_02) {
        draw_text(state);
        return;
    }
    if(event->wid == state->button_01) {
        draw_01_button(state);
        return;
    }
    if(event->wid == state->button_02) {
        draw_02_button(state);
        return;
    }
    if(event->wid == state->button_03) {
        draw_03_button(state);
        return;
    }
    if(event->wid == state->button_04) {
        draw_04_button(state);
        return;
    }
}

void handle_mouse_event(cstate *state)
{
    ～省略～
}

void handle_keyboard_event(cstate *state)
{
    ～省略～
}

void handle_event(cstate *state)
{
    ～省略～
}

void start_init(cstate *state)
{
    ～省略～
}

void init_gui(cstate *state)
{
    ～省略～
}

void wait_for_start(cstate *state)
{
    ～省略～
}

void s_start(cstate *state)
{
    while(state->state == STATE_START) {
        GrGetNextEventTimeout(&state->event,500);
        handle_event(state);
        state->ev_cnt++;
        if(state->dio_data) {
            draw_text(state);
            state->dio_data = 0;
        }
    }
}

void main_event_loop(cstate *state)
{
    ～省略～
}

void *main_dio(void *c){
cstate *state;
unsigned char buf[8];
    state = (cstate *)c;
    if(( state->fd = open("/dev/dio",O_RDWR)) == -1) {
        fprintf(stderr, "Can't open /dev/dio¥n");
        exit(2);
    }
    ioctl(state->fd,DIO_DRVR_INT_ENABLE);
    while(1) {
        read(state->fd,buf, 1);
        state->input_data = (int)buf[0];
        state->dio_data = 1;
        fprintf(stderr, "main_dio¥n");
    }
}

int main(int argc, char *argv[])
{
    cstate *state;
```

(a) cqsample.c

〔リスト8〕GUIプログラムのソース(つづき)

```
    if(!(state = malloc(sizeof(cstate)))) {
        fprintf(stderr, "Out of memory¥n");
        exit(1);
    }
    if(GrOpen() < 0) {
        fprintf(stderr, "Cannot Open Graphics¥n");
        exit(1);
    }
    init_gui(state);
    if(pthread_attr_init(&t_attr)) {
        fprintf(stderr, "pthread_attr_init error ¥n");
        exit(1);
    }
    if(pthread_attr_setinheritsched(&t_attr, INHERIT_SCHED)) {
        fprintf(stderr, "pthread_attr_setinheritsched error ¥n");
        exit(1);
    }
    if(pthread_attr_setschedpolicy(&t_attr, SCHED_RR)) {
        fprintf(stderr, "pthread_attr_setinheritsched error ¥n");
        exit(1);
    }
    if(pthread_create(&tid1, &t_attr, main_dio, state)) {
        fprintf(stderr, "pthread_create error ¥n");
        exit(1);
    }
    main_event_loop(state);
    GrClose();
    return 0;
}
```

(a) cqsample.c

```
#ifndef CQSAMPLE_H
#define CQSAMPLE_H
/*
*/
#define BORDER_WIDTH            10
#define BUTTON_HEIGHT       20
#define BUTTON_WIDTH            80
#define BUTTON_BACKGROUND_COLOUR LTGRAY
#define BUTTON_FOREGROUND_COLOUR BLACK
#define TEXT_X_POSITION         5
#define TEXT_Y_POSITION         15
#define TEXT2_Y_POSITION        30

#define MAIN_WINDOW_X_POSITION  100
#define MAIN_WINDOW_Y_POSITION  100
#define MAIN_WINDOW_WIDTH   320
#define MAIN_WINDOW_HEIGHT  240
#define MAIN_WINDOW_BACKGROUND_COLOUR CYAN

#define WINDOW_01_X_POSITION    200
#define WINDOW_01_Y_POSITION    80
#define WINDOW_01_WIDTH         100
#define WINDOW_01_HEIGHT        35
#define WINDOW_01_BACKGROUND_COLOUR BLACK
#define WINDOW_01_FOREGROUND_COLOUR GREEN

#define WINDOW_02_X_POSITION    100
#define WINDOW_02_Y_POSITION    80
#define WINDOW_02_WIDTH         100
#define WINDOW_02_HEIGHT        35
#define WINDOW_02_BACKGROUND_COLOUR WHITE
#define WINDOW_02_FOREGROUND_COLOUR GREEN

#define BUTTON_01_X_POSITION    10
#define BUTTON_01_Y_POSITION    80
#define BUTTON_01_WIDTH     BUTTON_WIDTH
#define BUTTON_01_HEIGHT    BUTTON_HEIGHT

#define BUTTON_02_X_POSITION    10
#define BUTTON_02_Y_POSITION    120
#define BUTTON_02_WIDTH     BUTTON_WIDTH
#define BUTTON_02_HEIGHT    BUTTON_HEIGHT

#define BUTTON_03_X_POSITION    100
#define BUTTON_03_Y_POSITION    180
#define BUTTON_03_WIDTH     BUTTON_WIDTH
#define BUTTON_03_HEIGHT    BUTTON_HEIGHT

#define BUTTON_04_X_POSITION    200
#define BUTTON_04_Y_POSITION    180
#define BUTTON_04_WIDTH     BUTTON_WIDTH
#define BUTTON_04_HEIGHT    BUTTON_HEIGHT

enum {
    STATE_START,
    STATE_STOP,
    STATE_EXIT,
    STATE_UNKNOWN
};

typedef GR_COLOR block;

struct cqsample_state {
    int ev_cnt;
    int input_data;
    int fhiscore;
    int level;
    int state;
    int old_state;
    int running_buttons_mapped;

    GR_WINDOW_ID main_window;
    GR_WINDOW_ID window_01;
    GR_WINDOW_ID window_02;

    GR_WINDOW_ID button_01;
    GR_WINDOW_ID button_02;
    GR_WINDOW_ID button_03;
    GR_WINDOW_ID button_04;

    GR_GC_ID maingcf;
    GR_GC_ID win_01_gcf;
    GR_GC_ID win_01_gcb;
    GR_GC_ID win_02_gcf;
    GR_GC_ID win_02_gcb;
    GR_GC_ID buttongcf;
    GR_GC_ID buttongcb;
    GR_EVENT event;
    int   fd;
    unsigned char dio_data;
};
typedef struct cqsample_state cstate;

void draw_text(cstate *state);
void draw_01_button(cstate *state);
void draw_02_button(cstate *state);
void draw_03_button(cstate *state);
void draw_04_button(cstate *state);

void job_02_button(cstate *state);
void job_03_button(cstate *state);
void job_04_button(cstate *state);

void handle_exposure_event(cstate *state);
void handle_mouse_event(cstate *state);
void handle_keyboard_event(cstate *state);
void handle_event(cstate *state);
void start_init(cstate *state);
void init_gui(cstate *state);
void wait_for_start(cstate *state);
void s_start(cstate *state);
void main_event_loop(cstate *state);
extern size_t strlen(char *buf);

#endif
```

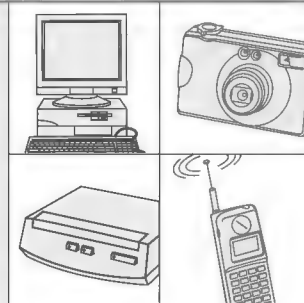
(b) cqsample.h

家電機器をネットワーク化するアーキテクチャ
Universal Plug and Play(UPnP)の全貌

第1回 UPnPの規格概要(前編)

茶間　康

従来，ネットワーク機器を接続するためには，IPアドレスの設定や各種デバイスドライバのインストールなど，煩雑な設定が必要だった．これを解決する手段として提案されたのがUniversal Plug and Play(UPnP)だ．UPnPを使用することにより，ネットワーク機器をPlug and Play感覚で使用することが可能になる．そしてその適応範囲はルータなどの純然たるネットワーク機器だけでなく，プリンタ，スキャナ，デジカメ，AV機器など，幅広い．また，基盤技術としてSOAP(Simple Object Access Protocol)とXMLを使用しているため，理解が容易である．

そこで本稿では，注目度が高まるUPnPについて連載で解説し，その理解を深める．第1回(今回)と第2回ではUPnPの規格について解説を行い，第3回では実際のUPnPプログラミングを解説する．(編集部)

2001年11月にWindows XPがリリースされ，Windows XPの機能の一つであるWindows Messengerが脚光を浴びました．そのときに必要とされたのが，Universal Plug and Play(以下UPnP)に対応したBroadbandルータ(UPnP Forumでは“Internet Gateway Device”というカテゴリになる)です．そのため，一般にはUPnP＝ルータといイメージがありますが，UPnPはそれ以外にも多くのことが可能な規格です．

Universal Plug and Playの目標

ネットワーク上にデバイスを接続するためには，PCと同じような機能が必要です．つまり，DHCPやDNSのようなサーバ機能が必要になりますし，デバイスの特徴にあった専用のプロトコルも必要になってきます．そのため，デバイスはかなり大規模な機能をもたなければならなくなるし，設定も面倒なものになります．これを改善するためには，Peer to Peerのネットワークにし，ユーザーがわずらわしい設定をしなくてよいPlug and Play方式を導入するという方法があります．

Plug and Playと聞くと，USBやPCカードをPCへ接続したときWindowsが自動的にデバイスを検出し，リソース(I/Oアドレス，DMA，割り込みポートなど，USBのときはパイプなど)を割り当て，デバイスドライバをロードし，ユーザーが簡単に使える機能を想像すると思います．UPnPは，これをネットワークの世界に拡張したものです．

● 現在のネットワーク周辺装置の問題

現在の周辺機器をネットワークに接続するときには，IPアドレス，サブネットマスク，デフォルトゲートウェイなどを設定しなければなりません．また，DHCPサーバがネットワーク上に存在し，このような設定を最小限にできたとしても，デバイスドライバやアプリケーションが，ターゲットの周辺機器のIPアドレスを手動で設定し，コントロールする必要が出てきます．このようなことを一気に解決するのがUPnPです．

UPnPによって，周辺機器はIPアドレスなどを自動的に割り振るのと同時に自らの機能や情報を通知し，また逆にアプリケーションはそれらの情報を検出できます．これにより，アプリケーションはターゲットの周辺機器を検出し制御できます．

今後，ネットワーク機器はPCゲーム機やハードディスクビデオレコーダなども含め，家庭で使われていくと思われます．このような場合，管理者がいないネットワークが構築されます．伝送メディアもワイヤレス，電力線，Ethernetなどさまざまになるでしょう．このような環境でもネットワーク機器が認識され，動作することをめざしているのがUPnP規格です．なお，UPnPはPeer to Peerネットワークです．

UPnPを構成するプロトコルスタック

UPnPデバイスアーキテクチャは，使用するプロトコルスタックを0から6までの機能(ステップ)で定義しています．これから，順を追って説明していきます．

UPnPは，インターネットで使われている既存のプロトコルを基礎としています．これにより，非常に実装しやすい規格になっています．その上に，UPnPデバイスアーキテクチャ(今回解説する部分)があります．

ここまでは，ネットワークに周辺機器がUPnP機器として参加するまでの基礎になります．実際には，これだけでは周辺機器を制御できません．コントロールするには，ビデオ，家電などのカテゴリごとに，どのように制御するのかを決めなければなりません．UPnP ForumではWorking Committee(WC)を設定し，規格(これをDCP：Device Control Protocolと呼んでいる)を定義しています．これは**図1**のUPnP Forumにあたります．UPnPベンダは，DCPの定義されている以上の機能をベンダが追加したいときに使われる部分になります．

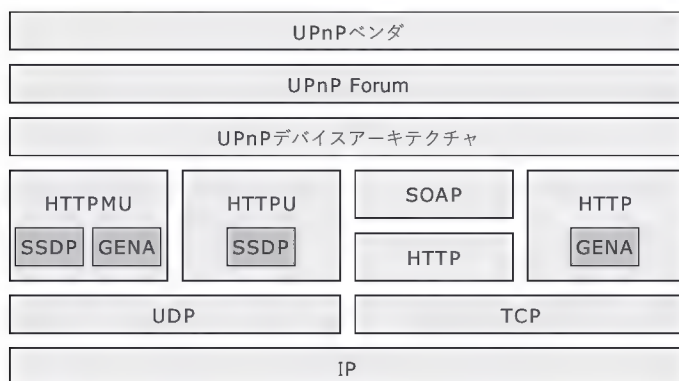
UPnPが使用する標準プロトコルの説明

● TCP/IP

TCP/IPは，UPnPが構築される基礎となります．UPnPはTCP/IPを採用したことにより，物理ネットワークメディアへの橋渡しをし，複数メディアの相互運用性を確保します．

UPnPデバイスはTCPやUDP，IGMP，ARP，IPなど

〔図1〕UPnPプロトコルスタック

〔図2〕UPnPネットワークコンポーネント

TCP/IPスタック内の多数のプロトコルを使用でき，DHCPやDNSなどのTCP/IPサービスを利用できます．今回は，TCP/IPについて読者が基本的に理解していると仮定しています．

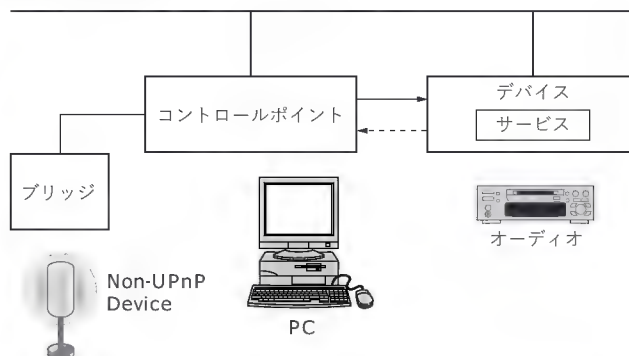
● HTTP，HTTPUおよびHTTPMU

HTTPも，UPnPの基礎となるプロトコルです．UPnPはHTTPとそのバリエーションの上に構築されています．

HTTPUおよびHTTPMUはHTTPを拡張したもので，TCP/IPではなくUDP/IPに基づいてメッセージを発信します．このプロトコルは，次に説明するSSDPでも使用されています．プロトコルで使用する基本メッセージ形式はHTTPであり，マルチキャスト通信およびオーバヘッドを必要としないメッセージ配信で必要になります．高レベルプロトコルおよびUPnPの動作の説明は，HTTPプロトコルについての基本的知識を前提としています．

● SSDP

Simple Service Discovery Protocol(SSDP)は，名前が表すようにネットワークサービス(デバイスとサービスの定義は，あいまいになりがちだが違うものである．決してデバイスだけを検出するわけではない．**図2**を参照のこと)を検出する方法を定義しています．SSDPはHTTPとHTTPMUに基づいて作成されていて，コントロールポイントが関心のあるサービス(リソース)を検索する方法と，デバイスがそれ自身のサービスをアナウンスする方法を定義しています．SSDPは検索要求とアナウンス方法を定義することにより，片一方のメカニズムだけでは回避できないオーバヘッドを軽減しています．その結果ネットワーク上のすべてのコントロールポイントは，ネットワークトラフィックを低くおさえながらネットワークの状態を正しく知ることができます．

コントロールポイントとデバイスはSSDPを使用します．UPnPコントロールポイントが開始されると，ネットワークで利用可能なデバイスとサービスを検出するため，HTTPMUを通してSSDP検索要求を送信できます．コントロールポイントは，たとえばVCRなど特定のタイプのデバイスだけを，あるいはクロックサービスなど特定のタイプのサービスだけ，あるいは特定のデバイスだけを検索するように調整できます．

UPnPデバイスはマルチキャストポートをリッスンします．デバイスが検索要求を受信すると，検索条件に一致するか調べます．一致していることがわかると，ユニキャストSSDP(HTTPU経由)レスポンスがコントロールポイントに送信されます．

同様にネットワークに追加されたデバイスは，サポートしているサービスを告知(アドバタイズ)するため，複数のディスカバリメッセージを送信します．

ディスカバリメッセージとユニキャストデバイスのレスポンスメッセージは，デバイスディスクリプションドキュメントのある場所のポインタを含んでいます．このドキュメントには，デバイスがサポートしているプロパティとサービスの情報があります．

またSSDPは，検出機能のほかにデバイスと関連サービスがネットワークからスマートに離脱する方法(byebyeの通知)を提供しています，古い情報を削除するキャッシュタイムアウトもこれに含まれています．

● GENA

Generic Event Notification Architecture(GENA：ジーナという)は，HTTP over TCP/IPおよびマルチキャストUDPを使用して通知を送受信する機能です．またGENAはイベントを可能にするため，通知のサブスクライバーとパブリッシャーのコンセプトを定義します．

UPnPではGENAフォーマットによってディスカバリメッセージを作成し，SSDPを使って送信します．これはサービス状態変更の情報を送ってUPnPイベントを発生するために使用します．イベント通知を受信する関連コントロールポイントは，関連サービスやイベントの送信場所，イベント通知のサブスクリプション時間を含む要求を送信してイベントソースをサブスクライブします．

サブスクリプションは，引き続き通知を受信するため定期的に更新される必要があり，またGENAを使ってキャンセルできます．

● SOAP

Simple Object Access Protocol(SOAP)は，リモートプロシージャコール(RPC)を実行するためのExtensible Markup

Language(XML)とHTTPの仕様を定義します．これは，RPCベースのインターネット通信の標準となっています．またSOAPは，セキュリティ用にセキュアソケットレイヤ(SSL)とHTTPの接続管理機能を使用でき，それによってWebページにアクセスするのと同じように簡単にインターネットを通して分散通信ができます．

UPnPはSOAPを使用して制御メッセージをデバイスに発信し，結果やエラー情報をコントロールポイントに返します．それぞれのUPnP制御要求は，パラメータに呼び出すアクションを指定しているSOAPメッセージです．レスポンスもSOAPメッセージを使用し，その中にステータスが含まれており，値とパラメータを返します．

● XML

W3Cの定義によれば，XMLはWeb上の構造化データ向けの汎用形式です．XMLは，さまざまな種類の構造化データをテキストファイルに入れることができます．

XMLは，タグと属性を使用する点でHTMLによく似ています．実際，このタグと属性の意味がグローバルに定義されていないことと，使用されてコンテキスト内で解釈されることとはまったく別です．XMLのこの特徴により，各種ドキュメントタイプに対してスキーマを開発するのに向いています．XMLのスキーマ言語としての仕様は，W3Cにより定義されています．

XMLは，デバイス/サービスディスクリプションやコントロールなどで使用されるUPnPの中心となります．

UPnPネットワークのコンポーネント

UPnPには，ネットワークの基本構成としてデバイス，サービス，コントロールポイントが定義されています．

デバイスはUPnPに対応した機器です．プリンタ，スキャナ，AV機器，家電などがそれにあたります．

サービスは，デバイスが提供する機能を表す最小単位です．たとえば，デバイスがオーディオプレーヤであれば，CDプレーヤサービスとクロックサービスなどが提供されます．音楽再生はプレイ/ストップ/スキップ/ポーズなどのアクションアイテムがあり，クロックサービスはゲットタイムやセットタイムなどのアクションアイテムをもっています．

コントロールポイントは，デバイスのもっているサービスを制御し，利用するものです．PCやスマートデバイスなどがそれにあたります．

デバイスは，最低一つのサービスをもっています．別の言い方をすると，オーディオプレーヤのようにCDプレーヤ，ラジオ，クロックという複数のサービスを一つのデバイスがもつことができます．デバイスの中に複数のデバイス(TVとVCRなどの複合機)をもたせることもできます．このとき，ベースになるデバイスをルートデバイスといいます．また，コントロールポイントとデバイスが一体になっている機器も考えられます．

なお，これは規格に定義されているわけではありませんが，UPnPは，UPnP以外のホームネットワークとの親和性を考慮して，その橋渡し機能を"ブリッジ"と定義しています．たとえば，ルームライトをON/OFFする程度でUPnPを実装するのは非常に高価なものになってしまいますが，通常は電灯線を利用するような簡単な方法(マイクロソフトはSimple Control Protocol：SCPという技術を提案している)を採用します．このように，UPnPではブリッジを介して通信する方法が検討されています．

UPnPの技術的な特徴

UPnPの技術的特徴を挙げると，以下の3点になります．

1) 物理的なネットワークメディアに依存しない
2) ソフトウェアプラットホームに依存しない
3) 標準インターネット技術を活用

1)について**図3**に示します．Ethernet，無線LAN，IEEE1394，PLC(電灯線)，HomeRF，HomePNAであっても，IPパケットを伝送できれば何でもかまいません．

2)は特定のOS，プログラム言語，APIなど，ソフトウェアに依存しないことです．これも重要です．これにより，メーカーはUPnPに参加している複数のベンダから提供されるチップセット，OSおよびソフトウェア開発キットを選択できます．

そして，もっとも重要な特徴は，3)のTCPやHTTP，XMLなど，すでに広く普及している既存の標準インターネットプロトコルやテクノロジを組み合わせて活用していることです．標準技術なので知識とスキルをもつ人が多く，実装も簡単です．そのため追加投資も少なくてすみ，またデバイスの間での接続性も保たれます．

UPnPのステップ(機能)

UPnPは，六つのステップによって実行されます．別の見方をすれば，六つの機能があります(**図4**)．

以降，この六つのステップについて解説します．

〔図3〕UPnPの技術的特徴

アドレッシング

UPnPは，IPアドレスを取得するためのプロトコルを定義していない間は，既存のプロトコルでIPアドレスを取得します．UPnPは規格をオープンなものにするため，意図的にオープンなプロトコルの上に構築されています．つまりアドレシッングに関しては，それらのルールを選択しているだけです．それで，アドレッシングをステップ0としているのでしょう．

各UPnPデバイスは，アドレスを得るためにDHCPクライアントの機能を含まなければなりません．DHCPがない場合には，IPネットワーク内に直接アドレスを割り当てるための規格があります．これは，IETFで定義されています．もし，DHCPサーバが利用可能でなければ，デバイスはIPアドレスを手動で割り当てなければなりません．

Auto-IPは，このような環境で簡単にIPアドレスを割り当てるためのプロトコルで，マイクロソフトによって定義されました．このドラフトは，IETFへ提案されました．Auto-IPを使用する場合，デバイスは169.254/16のレンジでアドレスをランダムに選択します．デバイスはこのアドレスが有効な場合，そのためのレスポンスを行います．これは，Address Resolution Protocol(ARP)packetを送信することにより完了します．もしアドレスが使用されていたならば新しいアドレスを再度選択し処理を続けます．加えてAuto-IPによってアドレスを取得した場合，デバイスは定期的にDHCPサーバが利用可能であるかどうかチェックしなければなりません．もし見つかった場合，デバイスはDHCPサーバから新しいアドレスを取得すべきです．

ディスカバリ

ディスカバリは，UPnPネットワーキングにおける最初のステップです．これは，ネットワーク機器を検出する方法を定義しています(**図5**)．Windows XPにデバイス(UPnP機器)を接続するとマイネットワークにアイコンが表示されますが，これはディスカバリ機能により実現しています．

ディスカバリは，以下の二つケースが考えられます．

1) デバイスがネットワークに追加される場合

UPnPディスカバリよりデバイスがネットワーク上のコントロールポイントに自分のサービスを告知(アドバタイズ)します．

2) コントロールポイントがネットワークに追加される場合

UPnPディスカバリにより，そのコントロールポイントがネットワーク上の関連デバイスを検索(サーチ)します．

どちらの場合も，UPnPはIPネットワーク上の他のデバイスを確認するために，デバイスまたはそのサービスに関するいくつかの必須事項(例：タイプ，識別子，詳細情報へのポインタなど)を含むディスカバリメッセージ(以降DsvMsgと記述)を使用します．

〔図4〕UPnPのステップ

3. コントロール	4. イベンティング	5. プレゼンテーション
2. ディスクリプション		
1. ディスカバリ		
0. アドレッシング		

〔図5〕ディスカバリ(アドバタイズ)の動作

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
CACHE-CONTROL: time to live
LOCATION: description url
NT: {UDN | type:version}
NTS: ssdp:alive
USN: UDN::NT header value
```

告知
(アドバタイズ)

〔図6〕UPnPデバイスの構成

SSDPでは，IPネットワーク上の機器の告知(アドバタイズ)，検出(サーチ)，削除(ディスコネクト)をするメカニズムを定義するのと同様に，DsvMsgをネットワークへマルチキャストするときに使用するマルチキャストアドレスとポート番号を定義しています．これはIANA(Internet Assigned Numbers Authority)がSSDPのために予約しています．そのため，このアドレス(239.255.255.250:1900)を使用する必要があります．

次に，ディスカバリを考えるときにデバイスの構成を考えなければなりません(**図6**)．UPnPは，デバイスの中に複数のデバイスやサービスを組み込むことができます．オーディオプレーヤを例にとると，ラジオ，CDプレーヤというデバイスを組み込んでいます．それぞれラジオは，チューナ，タイマというサービス，

CDプレーヤはそのままCDというサービスをもっています．また，オーディオプレーヤは電源管理，時計というサービスをもっています．UPnPでは，ここでのオーディオプレーヤはルートデバイスになります．

ディスカバリでは，ルートデバイスおよびすべての組み込みデバイス，サービスをコントロールポイントへアドバタイズする必要があります．また，コントロールポイントが，目的の(コントロールする)デバイスおよびサービスがサーチできる必要があります．**図7**にそのようすを表します．

では，ディスカバリ(SSDP)の動作をケースに分けて説明していきます．

- 告知(アドバタイズ)：UDP ── HTTPMU ── SSDP, GENA

デバイスが接続されたときに実行されます．ルートデバイス，組み込みデバイス，サービスを，各コントロールポイントへマルチキャストで告知します．そのときのディスカバリ(アドバタイズ)メッセージは，GENAおよびSSDPのメソッド，ヘッダにより構成されます．

SSDPはIP上のブロードキャストプロトコルです．SSDPはUDP上で動作する二つのHTTPに依存しています．これらは，HTTPを拡張したものでHTTPMU/HTTPUといい，SSDPの一部として定義されています．アドバタイズメッセージは，HTTPMU/HTTPUを介して送信されます(**図8**)．

それぞれどのような情報をどのように告知するか説明します．コントロールポイントは告知のメッセージを受け取ることでデバイスを確認し，ステップ2のディスクリプションへ移ります．アドバタイズで使用するDsvMsgの形式を**表1**に示します．

〔図7〕ディスカバリメッセージ動作

▶ルートデバイス

三つのDsvMsgをコントロールポイントへマルチキャストします．UPnPは，階層化されたデバイス構成をサポートします．つまり，コンテナとしての役割があります．ルートデバイスの機能をアドバタイズするだけでなく，その中に組み込まれた各デバイスもアドバタイズする必要があります．ルートデバイス(コンテナ)のために送られるDsvMsgから始まります．`upnp:rootdevice`のNTヘッダをもつスペシャルメッセージが，コンテナとしてのデバイスを認識させるため送信されます．

また，各デバイスはネットワーク上では，Unique Device Name(UDN)によって表現されます．UDNはデバイスを包括的にユニークな値をもつUniversally Unique IDentifier(UUID)で指定します．つまり，UPnPはネットワーク内でのデバイスをUUIDで識別するわけです．

通知パケットはこのUUIDを含んだものが送信されます．三つのDsvMsgは，それぞれ`NT: upnp:rootdevice`，`NT:root device UUID(NT: root device UDN)`および`NT:device type:version`です．USNとの組み合わせを**表2**に示します．

▶組み込みデバイス

組み込みデバイスは，二つのDsvMsgをコントロールポイントにマルチキャストします．`NT: embedded device UUID(NT: embedded device UDN)`，`NT: device type:version`です．USNとの組み合わせを**表3**に示します．

▶サービス

サービスタイプとバージョン(`NT: device type:version`)のDsvMsgを1回だけコントロールポイントにマルチキャストします．これによりコントロールポイントがコントロールできるデバイス内のサービスを検出できます．サービスはUDNをもっていないので，サービスを提供するデバイスのUDNをDsvMsg内で使用します．USNとの組み合わせを**表4**に示します．

以上の組み合せを考慮すると“ディスカバリメッセージ動作でのアドバタイズ”の例(**図7**)では，告知においてDsvMsgを7回送信することになります．

〔図8〕ディスカバリ(アドバタイズ)において使用するプロトコル

〔表1〕ディスカバリーーアドバタイズメッセージ

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
CACHE-CONTROL: max-age = seconds until advertisement expires
LOCATION: URL for UPnP description for root device
NT: search target
NTS: ssdp:alive
SERVER: OS/version UPnP/1.0 product/version
<SPAN class=ssdp>USN: advertisement UUID
<BR>
```

(NOTIFYメソッドのリクエストにはボディはないが，メッセージには最後のHTTPヘッダの後にブランク行が必要)

リクエストライン		
NOTIFY	GENAにより定義されたメソッドで，通知やイベントに使用する．ディスカバリはデバイスのアドバタイズのため使用する	
*	リクエストは個々のリソースではなく全体に適用される．*でなくてはならない	
HTTP/1.1	HTTPバージョン	
ヘッダ		
HOST	必須項目．IANAによりSSDPのために予約されたマルチキャストチャネルおよびポート．239.255.255.250:1900でなくてはならない	
CACHE-CONTROL	必須項目．通知が有効な秒数を指定する．この期限を過ぎると，コントロールポイントはデバイス(またはサービス)が無効になったと解釈する．1800秒(30分)より長く設定する必要がある．UPnPベンダにより指定される．整数	
LOCATION	必須項目．ルートデバイスのUPnPディスクリプションへのURLを含む．管理されていないネットワークでは，このURLのホストにはIPアドレス(ドメイン名に対し)が含まれる場合がある．UPnPベンダにより指定される．単一のURL	
NT	GENAが定義する必須ヘッダ．Notification Type(通知タイプ)の頭文字．以下のいずれかでなくてはならない．単一のURI(Uniform Resource Identifier)	
	UPnP:rootdevice	ルートデバイスに対し1度送信
	uuid:device-UUID	各デバイス* に対し1度送信．UDNを指定する**
	urn:schemas-UPnP-org: device:deviceType: v	各デバイス* に対し1度送信***
	urn:schemas-UPnP-org: service:serviceType: v	各サービスに対し1度送信****
NTS	GENAが定義する必須ヘッダ．Notification Sub Type(通知サブタイプ)の頭文字．ssdp:aliveでなくてはならない．単一のURI	
SERVER	必須項目．OS名，OSバージョン，UPnP/1.0，製品名および製品バージョンを連記したもの．UPnPベンダにより指定される．文字列	
USN	SSDPが定義する必須ヘッダ．ユニークサービス名．以下のいずれかでなくてはならない．接頭辞("::"の前)は，デバイスディスクリプションのUDNエレメントの値と一致していなくてはならない(ディスクリプションのセクションで，UDNエレメントについて説明している)．単一のURI	
	uuid:device-UUID:: UPnP:rootdevice	ルートデバイスに対し1度送信**
	uuid:device-UUID	各デバイス* に対し1度送信**
	uuid:device-UUID::urn: schemas-UPnP-org:device: deviceType:v	各デバイス* に対し1度送信** ***
	uuid:device-UUID::urn: schemas-UPnP-org:service: serviceType:v	各サービスに対し1度送信** **** (NOTIFYメソッドによるリクエストに対する応答はない)

*：各デバイスはルートまたは組み込み．　**：デバイスUUID(UDNのこと)はUPnPベンダにより指定される．
***：デバイスタイプおよびバージョンはUPnP Forumワーキングコミッティにより定義される．
****：サービスタイプおよびバージョンはUPnP Forumワーキングコミッティにより定義される．

〔表2〕ルートデバイスアドバタイズメッセージの組み合わせ

	NT	USN
1	root device UUID(UDN)	root device UUID(UDN)
2	device type : device version	root device UUID :: device type : device version
3	UPnP:rootdevice	root device UUID :: UPnP:rootdevice

〔表3〕組み込みデバイスアドバタイズメッセージの組み合わせ

	NT	USN
1	embedded device UUID(UDN)	embedded device UUID(UDN)
2	device type : device version	embedded device UUID :: device type : device version

〔表4〕サービスアドバタイズメッセージの組み合せ

	NT	USN
1	Service type : service version	device UUID(no service):: service type : service version

● 削除(ディスコネクト)：UDP ── HTTPMU ── SSDP, GENA

削除も告知の一つと考えられますが，今回は違うものとして考えます．

デバイスおよびそのサービスがネットワークから削除されようとし(電源を切るときなど)，告知のDsvMsg(NTS： ssdp:alive)の期限が切れていない(CACHE-CONTROLの値を越えていない)場合には，削除のDsvMsg(NTS： ssdp:byebye)をマルチキャスト送信します．デバイスが送信したssdp:aliveメッセージすべてに対してssdp:byebyeメッセージを送信することにより，デバイスの削除が完了します．

デバイスがネットワークから不意に除外されると，メッセージをマルチキャスト送信できない場合があります．それに備えDsvMsgには，(すでに説明したとおり)CACHE-CONTROLヘッダに有効期限を設定します．再通知されない場合，ディスカバリメッセージは期限切れとなり，すべてのコントロールポイントから削除されます．つまりCACHE-CONTROL以内に告知を再度行わないと，コントロールポイントはそのデバイスを認識しなくなります．

なお，削除のときにもNOTIFYメソッドを使用します．違いは，NTSがbyebyeであることです．

余談ですが，よく質問されることなので説明すると，DHCPサーバがネットワーク上に接続され，Auto-IPで割り振られたIPアドレスを変更しなければならないことがあります．このときデバイスがどのようにすればいいかを説明します．まず，byebyeでデバイスを削除し，ネットワークからいったん，論理的に切り離します．その後アドレッシングでDHCPからIPアドレスを取得し，ディスカバリ，ディスクリプションとネットワークに論理的に接続していきます．このとき，まわりのコントロールポイントやデバイスは，新しいIPアドレスにすでに変わっているかもしれませんが，UPnPではデバイスをIPアドレスではなくUDN(UUID)で管理しているため識別することができます．では，削除で使用するDsvMsgの形式を**表5**に示します．

〔**表5**〕**ディスカバリーーディスコネクトメッセージ**

NOTIFY * HTTP/1.1 HOST: 239.255.255.250:1900 < SPAN class=gena>NT: search target NTS: ssdp:byebye < SPAN class=ssdp>USN: advertisement UUID (NOTIFYメソッドのリクエストにはボディはないが，メッセージには最後のHTTPヘッダの後にブランク行が必要) ディスコネクト時のディスカバリメッセージのリクエストライン，ヘッダは，アドバタイズのときの説明と同じ．使用しないヘッダもあるが内容は同じ． 違うヘッダはNTSだけである	
NTS	GENAが定義する必須ヘッダ．Notification Sub Type(通知サブタイプ)の頭文字．ssdp:byebyeでなくてはならない．単一のURI

〔**図9**〕**ディスカバリ**(サーチ)**の動作**

● 検索(サーチ)

コントロールポイントがネットワークに接続されたときには，コントロールできるデバイスをネットワーク上から探してくる必要があります．それが検索です(**図9**)．デバイス/サービスの種類やバージョン/デバイス名(UDN)を指定してデバイスを検索できます．検索のためのDsvMsgは，マルチキャストする必要があります．

検索されたデバイスは"サーチレスポンス"という応答をします．このときのDsvMsgは，サーチしたコントロールポイントだけにユニキャストします．

このDsvMsgは，SSDPのメソッド，ヘッダにより構成されます．**図10**のとおり，GENAはコントロールポイントがデバイスを検索するときには使われません．

▶サーチ：UDP ── HTTPMU ── SSDP

UPnPでは，三つの検索方法を用意しています．すべてのデバイスを検出(All)，type/versionによる検出，そしてデバイスを指定した検出です．このディスカバリ(サーチ)メッセージでは，新しいHTTPのM-SEARCHメソッドを使用します．サーチメッセージの形式を**表6**に示します．

▶サーチレスポンス：UDP ── HTTPU ── SSDP

コントロールポイントのサーチにレスポンスするため，デバイスはソースIPアドレスおよびポートにレスポンスメッセージを送信します．つまりサーチの発送元にIPアドレスにレスポンス

〔**図10**〕**ディスカバリ**(サーチ)**において使用するプロトコル**

〔表6〕ディスカバリーーサーチメッセージ

```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
MX: time to delay
ST: {ssdp:all | UDN | type:version}
<BR>
```

(M-SEARCHメソッドのリクエストにはボディはないが，メッセージには最後のHTTPヘッダの後にブランク行が必要)

リクエストライン		
M-SEARCH	SSDPにより定義されたサーチリクエストのメソッド	
*	リクエストは個々のリソースではなく全体に適用される．*でなくてはならない	
HTTP/1.1	HTTPバージョン	
ヘッダ		
HOST	必須項目．IANAによりSSDPのために予約されたマルチキャストチャネルおよびポート．239.255.255.250:1900でなくてはならない	
MAN	必須項目．NTSおよびSTヘッダと異なり，MANヘッダの値はダブルクォーテーションで括られる．必ず"ssdp:discover"でなくてはならない	
MX	必須項目．最大待機時間．デバイスの応答は0からこの設定値までのランダムな時間分遅延することにより，コントロールポイントがレスポンス処理する際の負荷を軽減する．レスポンスするデバイスの数が多くなる場合や，ネットワーク遅延があるときなどは，この値を増加させる．UPnPベンダにより指定される．整数	
ST	SSDPが定義する必須ヘッダ．Search Target(検索ターゲット)の頭文字．以下のいずれかでなくてはならない．単一のURI	
	ssdp:all	すべてのデバイスおよびサービスを検索
	UPnP:rootdevice	ルートデバイスのみ検索
	uuid:device-UUID	特定のデバイスを検索．UDNを指定する*
	urn:schemas-UPnP-org: device:deviceType: v	指定したタイプのデバイスをすべて検索**
	urn:schemas-UPnP-org: service:serviceType: v	指定したタイプのサービスをすべて検索***

*：UDNは，UUIDによってUPnPベンダにより指定される．
**：デバイスタイプおよびバージョンはUPnP Forumワーキンググコミッティにより定義される．
***：サービスタイプおよびバージョンはUPnP Forumワーキンググコミッティにより定義される．

のDsvMsgを送信します．コントロールポイントはレスポンスのメッセージを受け取ることでデバイスを確認し，ステップ2のディスクリプションに移ります．なお，このDsvMsgは，告知時のメッセージと似ています．サーチレスポンスメッセージの形式を**表7**(次頁)に示します．

サーチのDsvMsgのMANヘッダが"ssdp:discover"でなかった場合，エラーになります．この場合，エラーが発生したことをコントロールポイントに返す必要があります．デバイスはHTTP error 412 Precondition Failedとレスポンスしてください．

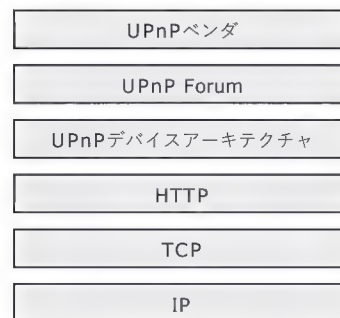
以上がディスカバリの内容です．なおUDPによる転送は，データの信頼性がTCPと比べると低くなります．そのため，DsvMsgをそれぞれ数回にわたって送信する必要がある場合もあります．

ディスクリプション

コントロールポイントがデバイスをディスカバリ(検出)しても，デバイスに関する情報はDsvMsgの内容しか得られません．つまり，デバイスまたはサービスのタイプやバージョン，UUIDによるデバイス名だけです．

ステップ2のディスクリプションでは，コントロールポイントが制御などを行うための詳細な情報をデバイスから取得します．

〔図11〕ディスクリプションにおいて使用するプロトコル

UPnPベンダ
UPnP Forum
UPnPデバイスアーキテクチャ
HTTP
TCP
IP

詳細は後述しますが，デバイスの機種名，メーカー，サービスの機能，コントロールのためのURLやコマンドがXMLで記述され，HTTPによって送信されます．

まず，コントロールポイントがデバイスからどのようにディスクリプションドキュメント(規格書ではディスクリプションドキュメントという言葉を使っていない)を取得するかを説明します．**図11**がディスクリプションで使用するプロトコルです．

ここで，ディスカバリを思い出してください．告知やサーチレスポンスのメッセージにLOCATIONヘッダがあったと思います．ここにはルートデバイスのディスクリプションドキュメントのURLが設定されています．コントロールリクエストは，このURLへHTTP GETリクエストを発行します．このリクエストの

〔表7〕ディスカバリーーサーチレスポンス

<table>
<tr><td colspan="3">HTTP/1.1 200 OK
CACHE-CONTROL: time to live
DATE: date generated
EXT:
LOCATION: description url
SERVER: OS/version UPnP/1.0 product/version
ST: same as NT header
USN: UDN::ST header value

(M-SEARCHメソッドのリクエストに対するレスポンスのディスカバリメッセージはボディがないが，メッセージには最後のHTTPヘッダの後にブランク行が必要)</td></tr>
<tr><td colspan="3">ヘッダ</td></tr>
<tr><td>CACHE-CONTROL</td><td colspan="2">必須項目．通知が有効な秒数を指定する．この期限を過ぎると，コントロールポイントはデバイス(またはサービス)が無効になったと解釈する．1800秒(30分)より長く設定する必要がある．UPnPベンダにより指定される．整数</td></tr>
<tr><td>DATE</td><td colspan="2">推奨項目．レスポンスのディスカバリメッセージが生成された日付を指定する．RFC 1123で定義されている日付と時間コードで指定する</td></tr>
<tr><td>EXT</td><td colspan="2">必須項目．MANヘッダの値を理解したかどうかの確認(ヘッダのみ．値なし)</td></tr>
<tr><td>LOCATION</td><td colspan="2">必須項目．ルートデバイスのUPnPディスクリプションへのURLを含む．管理されていないネットワークでは，このURLのホストにはIPアドレス(ドメイン名に対し)が含まれる場合がある．UPnPベンダにより指定される．単一のURL</td></tr>
<tr><td>SERVER</td><td colspan="2">必須項目．OS名，OSバージョン，UPnP/1.0，製品名および製品バージョンを連記したもの．UPnPベンダにより指定される．文字列</td></tr>
<tr><td rowspan="6">ST</td><td colspan="2">SSDPが定義する必須ヘッダ．Search Target(検索ターゲット)の頭文字．サーチのディスカバリメッセージのSTの値によりレスポンスの動作が変わる．単一のURI</td></tr>
<tr><td>ssdp:all</td><td>アドバタイズのディスカバリメッセージのときと同じように，ルートデバイスは3回，一つの組み込みデバイスは2回，組み込みサービスは1回である．組み込みデバイスがd個，サービスがk個あればルートデバイスは，$3+2d+k$回レスポンスしなければならない．図7の例だとレスポンスにおいてディスカバリメッセージを7回送信する．STヘッダの値は，アドバタイズ(ssdp:aliveのNOTIFY)メッセージのNTヘッダの値をそれぞれのレスポンスのディスカバリメッセージで指定しなければならない．単一のURI</td></tr>
<tr><td>UPnP:rootdevice</td><td>ルートデバイスに対し1度レスポンス．単一のURI</td></tr>
<tr><td>uuid:device-UUID</td><td>各デバイス* に対し1度レスポンス．UDNを指定しなければならない.** 単一のURI</td></tr>
<tr><td>urn:schemas-UPnP-org: device:deviceType: v</td><td>各デバイス* に対し1度レスポンス***</td></tr>
<tr><td>urn:schemas-UPnP-org: service:serviceType: v</td><td>各サービスに対し1度レスポンス****</td></tr>
<tr><td>USN</td><td colspan="2">SSDPが定義する必須ヘッダ．ユニークサービス名．以下のいずれかでなくてはならない．接頭辞(“::”の前)は，デバイスディスクリプションのUDNエレメントの値と一致していなくてはならない(ディスクリプションのセクションで，UDNエレメントについて説明している)．単一のURI．内容はアドバタイズのディスカバリメッセージのUSNの項目を参照してほしい</td></tr>
</table>

*：各デバイスはルートまたは組み込み．　**：デバイスUUID(UDNのこと)はUPnPベンダにより指定される．
***：デバイスタイプおよびバージョンはUPnP Forumワーキングコミッティにより定義される．
****：サービスタイプおよびバージョンはUPnP Forumワーキングコミッティにより定義される．

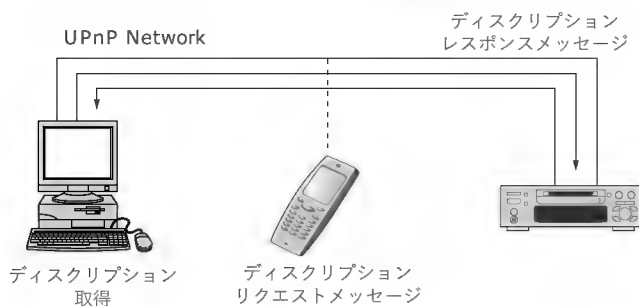
〔表8〕ディスクリプションーーリクエストメッセージ

<table>
<tr><td colspan="3">GET path to description HTTP/1.1
HOST: host for description:port for description
ACCEPT-LANGUAGE: language preferred by control point

(ディスクリプションを取得するためのメッセージにはボディはないが，メッセージの最後のHTTPヘッダの後にブランク行が必要)</td></tr>
<tr><td colspan="3">リクエストライン</td></tr>
<tr><td rowspan="3">GET</td><td colspan="2">HTTPにより定義されたメソッド</td></tr>
<tr><td>path to description</td><td>デバイスディスクリプションのURL(ディスカバリメッセージのLOCATIONヘッダ)またはサービス記述URL(デバイスディスクリプションのSCPDURLエレメント)．単一の相対URL</td></tr>
<tr><td>HTTP/1.1</td><td>HTTPバージョン</td></tr>
<tr><td>ヘッダ</td><td colspan="2"></td></tr>
<tr><td>HOST</td><td colspan="2">必須項目．デバイスディスクリプションURLのドメイン名またはIPアドレス，およびオプションのポート番号(ディスカバリメッセージのLOCATIONヘッダ)またはサービスディスクリプションURL(デバイス記述のSCPDURLエレメント)．ポートが指定されていない場合，あるいは明記されていない場合，80が用いられる</td></tr>
<tr><td>ACCEPT-LANGUAGE</td><td colspan="2">推奨項目．コントロールポイントが希望するディスクリプションに使用される言語．指定した言語で記述されたディスクリプションがない場合，デバイスはデフォルト言語のディスクリプションを返す．RFC 1766の言語タグ</td></tr>
</table>

〔図12〕ディスクリプションリクエストの動作

〔図13〕デバイスディスクリプションの簡略図

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
<device>
   <iconList>
<icon>
</icon>
</iconList>
<serviceList>
<service>
</service>
</serviceList>
<deviceList>
            <device>   ←入れ子になる
                    <serviceList>
</deviceList>
</device>
</root>
```

デバイスからのレスポンスのボディにデバイスディスクリプション(以降DevDescと記述する)が含まれます．このリクエストおよびレスポンスメッセージはHTTP，TCP/IPを介して配信されます．またサービスに関するディスクリプションは，DevDesc内に記述されているURLから取得します．**図12**にコントロールポイントがDevDescを取得する動作を示します．

● ディスクリプションリクエスト

コントロールポイントは，GETメソッドのメッセージ(リクエスト)を送信します．**表8**にディスクリプションのリクエストメッセージフォーマットを示します．

● ディスクリプションレスポンス

コントロールポイントがリクエストを送信したら，デバイスは次のステップに進み，ディスクリプションを返します．デバイスは予定転送時間を含め30秒以内にレスポンスしなければなりません．この時間内にレスポンスできなかった場合，コントロールポイントはリクエストを再送します．デバイスは必ず**表9**のメッセージでレスポンスを送信します．レスポンスメッセージのボディには，デバイス/サービスのディスクリプションが入ります．

● ディスクリプションのタイプ

ディスクリプションには大きく二つの種類があります．一つは，デバイスの構成などを示すDevDesc，もう一つは実際にデバイスが実行する機能を示すサービスディスクリプション(以降SerDescと記述する)です．DevDescは，デバイスというコンテナ(箱)を表すため，あえて分けると三つのパートに分けることができます．

1) 物理プロパティ
2) 論理プロパティ
3) UIプロパティ

物理プロパティはモデル名やモデル番号，製造シリアル番号，メーカー名，メーカーのWebページへのURLなど，メーカー情報が含まれます．論理プロパティはサービスタイプ，名前，SerDescへのURL，コントロールへのURL，そしてイベンティングへのURLが含まれます．後の項目でコントロール，イベンティングに関して詳しく説明します．UIプロパティはIEなどのWebブラウザ経由でデバイスを設定，コントロールするためのURLを示すものです．これはプレゼンテーションと呼ばれます．

DevDescには，すべての組み込みデバイスのDevDescも`<device list>`というエレメントに入れ子状に表現されます．**図13**にデバイスディスクリプションで組み込みデバイスを表現する方法を示します．

SerDescは，コントロールポイントからのリクエストに答えるコマンドまたはアクションの一覧と，各アクションに対するパラメータまたは引き数が含まれます．SerDescには変数の一覧

〔表9〕ディスクリプション――レスポンスメッセージ

```
HTTP/1.1 200 OK
CONTENT-LANGUAGE: language used in description
CONTENT-LENGTH: Bytes in body
CONTENT-TYPE: text/xml
DATE: when responded
<BODY>
Description
        レスポンスメッセージのボディには，デバイス/サービスのディスクリプションが入る
```

ヘッダ	
CONTENT-LANGUAGE	リクエストにACCEPT-LANGUAGEヘッダが含まれる場合のみ必須項目．ディスクリプションを記述する言語．RFC 1766の言語タグ
CONTENT-LENGTH	必須項目．バイト単位で示したボディの長さ．整数
CONTENT-TYPE	必須項目．text/xmlでなくてはならない
DATE	推奨項目．レスポンスが生成された日付．RFC 1123の日付

も含まれます．これらの変数は，サービスの実行時の状態をモデル化するもので，データ型や範囲およびイベント特性などにより記述されます．このセクションではアクション，引き数，状態変数そしてこれらの変数のプロパティの記述について説明しています．イベント特性については，イベンティングのセクションで説明しています．

DevDesc，SerDescともに記述にはXML構文をUPnPのため拡張（派生）したUPnP Template Languageを使用します．通常，各UPnP WC（Working Committee）で策定される各DCP（デバイスコントロールプロトコル）のUPnPデバイステンプレート，UPnPサービステンプレートにベンダ固有の情報（モデル名やモデル番号，製造シリアル番号など）を追加することでDevDesc，SerDescになります．現在，IGD，AV，Printer，Scannerなどの規格が策定されています（http://www.upnp.org/standardizeddcps/default.asp）．

● メーカー固有の機能

では，メーカーが他社のデバイスと差別化することはできないのでしょうか？　もちろんできます．メーカーは標準デバイスへ追加のUPnPサービスを含めたり，追加デバイスを組み込んだりして機能を拡張することができます．ただし，DCPが標準化されているものであれば必須の機能はサポートし，そこへ機能強化すべきです．こうすることで，コントロールポイントや他のデバイスとの最低限の互換性が保たれます．また，規格化されていないデバイスを検討されているときには，まずUPnPフォーラムでの話し合いを検討されることをおすすめします．

UPnP Template Languageの詳細に関しては，ここでは説明しません．UPnP Device Architecture V1.0をご参照ください．

● デバイスディスクリプション

ここでは，DevDescのエレメントを示します．これにより，各WCの規格を理解できると思います．エレメントの順序は重要ではありません．明記されていないかぎり，必須エレメントは重複することはなく1度だけ発生し，推奨あるいは任意エレメントは発生しない場合もありますが，発生する場合は1度だけとなります．**表10**(pp.187-189)にDevDescの形式を示します．

UPnPフォーラムのページ（http://www.upnp.org/resources/devices.asp）でDevDescの例からCD Playerの中の一部分を示します（**リスト1**．なお，この例は正式にDCP規格になったものではなく，あくまでサンプル）．このデバイスのデバイスタイプはCDPlayer:1と定義されています．

また，このデバイスは，SwitchPower，ChangeDisc，Play CD，Audioというサービスをもっています．このデバイスの設計者は，URL to service description，URL for control，URL for eventingを設定すればDevDescを作成できます（**リスト2**）．

● サービスディスクリプション

DevDescに記述されるSerDescのURLへHTTP GETすることでSerDescを取得できます．ここでは，この内容を説明します．

SerDescは，アクションとその引き数，そしてステートバリアブル（状態変数）とそのデータ型，範囲，イベント特性を定義するものです．各サービスにはゼロまたはそれ以上のアクションがあります．各アクションにはゼロまたはそれ以上の引き数があります．これらの引き数の組み合わせが，入力/出力パラメータとなります．アクションに一つ以上の出力引き数がある場合，

〔リスト1〕CDプレーヤのデバイスディスクリプションの例

```
<device>
    <deviceType> urn:schemas-upnp-org:device:CDPlayer:1
                                                    </deviceType>
    <friendlyName>short user-friendly title</friendlyName>
    <manufacturer>manufacturer name</manufacturer>
```

〔リスト3〕Audioのサービステンプレートの例

```
<action> <!-- absolute set -->
      <name>SetVolume</name>
      <argumentList>
        <argument>
          <name>NewVolume</name>
          <relatedStateVariable>Volume</relatedStateVariable>
          <direction>in</direction>
        </argument>
     </argumentList>
    </action>
.
.
  <serviceStateTable>
    <stateVariable sendEvents="yes">
      <name>Volume</name>
      <dataType>ui1</dataType>
      <allowedValueRange>
        <minimum>0</minimum>
        <maximum>255</maximum>
      </allowedValueRange>
    </stateVariable>
```

〔リスト2〕CDプレーヤのデバイスディスクリプションのサービス部の例

```
<service>
        <serviceType>urn:schemas-upnp-org:service:SwitchPower:1
                                                      </serviceType>
        <serviceId>urn:upnp-org:serviceId:SwitchPower</serviceId>
        <SCPDURL>URL to service description</SCPDURL>
        <controlURL>URL for control</controlURL>
        <eventSubURL>URL for eventing</eventSubURL>
      </service>
      <service>
        <serviceType>urn:schemas-upnp-org:service:ChangeDisc:1
                                                      </serviceType>
        <serviceId>urn:upnp-org:serviceId:ChangeDisc</serviceId>
        <SCPDURL>URL to service description</SCPDURL>
        <controlURL>URL for control</controlURL>
        <eventSubURL>URL for eventing</eventSubURL>
      </service>
      <service>
        <serviceType>urn:schemas-upnp-org:service:PlayCD:1
                                                      </serviceType>
        <serviceId>urn:upnp-org:serviceId:PlayCD</serviceId>
        <SCPDURL>URL to service description</SCPDURL>
        <controlURL>URL for control</controlURL>
        <eventSubURL>URL for eventing</eventSubURL>
      </service>
      <service>
        <serviceType>urn:schemas-upnp-org:service:Audio:1
                                                      </serviceType>
        <serviceId>urn:upnp-org:serviceId:Audio</serviceId>
        <SCPDURL>URL to service description</SCPDURL>
        <controlURL>URL for control</controlURL>
        <eventSubURL>URL for eventing</eventSubURL>
      </service>
```

これら引き数は，コントロールの結果として返り値として使われます．各引き数は，一つの状態変数に対応します．各サービスには一つ以上の状態変数があります．

また，非標準デバイスの定義に加え，UPnPベンダは標準デバイスにアクションやサービスを追加することができます．**表11**(pp.190-192)にSerDescの形式を示します．

DevDescの例で示したCD Playerは，SwitchPower，Change Disc，PlayCD，Audioのサービスをもっています．フォーラムの中でChangeDisc，PlayCD，Audioのサービステンプレートを示していますが，今回はAudioの一部分を示します(**リスト3**．なお，この例は正式にDCP規格になったものではなく，あくまでサンプル)．これでいくとSetVolumeという音量を変更するアクションがあります．その状態変数はVolumeで0から255までの設定ができます．なお，この変数の型はui1となっているので1バイト整数型になります．コントロールポイントは，たとえば53という値でこのアクションを実行すると，ボリュームが53を示すことになります．

最後になりますが，サービスがアクションをもたず，状態変数やイベンティングをもつことは，論理的には可能です．ただし，サービスが状態変数をもたないことはあり得ません．

● ベンダ拡張による非標準

UPnPベンダは，追加サービスや追加デバイスを組み込むことで各自のデバイスを差別化し，標準デバイスを拡張することができます．同様に，UPnPベンダはアクションや状態変数を追加することで標準サービスを拡張することができます．ただしUPnPベンダは，標準化されたallowedValueListを修正して標準サービスを拡張することはできません．これを行うと互換性に問題を与えるためです．拡張するための，それぞれの名前の慣習については，**表12**(p.192)にまとめます．

まとめ

Universal Plug and Play Device Architecture V1.0のスペックの話をしてきましたが，詳細に関してはスペックそのものを参照してください．これまでの話でUPnPデバイスアーキテクチャV1.0や各DCPのスペックを理解する上での助けになると思います．

今回は六つのステップのうち三つを解説しました．次回は，コントロール以降についての解説を行います．

参考文献

1) Universal Plug and Play Device Architecture V1.0，http://www.upnp.org/download/UPnPDA10_20000613.htm
2) Simple Service Discovery Protocol(SSDP)，http://www.upnp.org/download/draft_cai_ssdp_v1_03.txt
3) Multicast and Unicast UDP HTTP Messages，http://www.upnp.org/download/draft-goland-http-udp-04.txt
4) General Event Notification Architecture(GENA)，http://www.upnp.org/download/draft-cohen-gena-client-01.txt
5) Sample Device and Service Templates，http://www.upnp.org/resources/devices.asp
6) Windows XPのUniversal Plug and Play，http://www.microsoft.com/japan/windowsxp/pro/techinfo/planning/upnp/UniversalPlugandPlayinWindowsXP.doc
7) Simple Object Access Protocol(SOAP)，http://www.w3.org/TR/SOAP/，http://www.microsoft.com/japan/developer/workshop/xml/general/soapspec.asp
8) Extensible Markup Language(XML)，http://www.w3.org/XML/

ちゃま・やすし テクニカルライター

〔**表10**〕**デバイスディスクリプション**

```
<?xml version="1.0"?>
<root xmlns="urn:schemas-upnp-org:device-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <URLBase>base URL for all relative URLs</URLBase>
  <device>
    <deviceType>urn:schemas-upnp-org:device:deviceType:v</deviceType>
    <friendlyName>short user-friendly title</friendlyName>
    <manufacturer>manufacturer name</manufacturer>
    <manufacturerURL>URL to manufacturer site</manufacturerURL>
    <modelDescription>long user-friendly title</modelDescription>
    <modelName>model name</modelName>
    <modelNumber>model number</modelNumber>
    <modelURL>URL to model site</modelURL>
    <serialNumber>manufacturer's serial number</serialNumber>
    <UDN>uuid:UUID</UDN>
    <UPC>Universal Product Code</UPC>
    <iconList>
      <icon>
        <mimetype>image/format</mimetype>
        <width>horizontal pixels</width>
        <height>vertical pixels</height>
```

〔表10〕デバイスディスクリプション(つづき)

```
        <depth>color depth</depth>
        <url>URL to icon</url>
      </icon>
      XML to declare other icons, if any, go here
    </iconList>
    <serviceList>
      <service>
        <serviceType>urn:schemas-upnp-org:service:serviceType:v</serviceType>
        <serviceId>urn:upnp-org:serviceId:serviceID</serviceId>
        <SCPDURL>URL to service description</SCPDURL>
        <controlURL>URL for control</controlURL>
        <eventSubURL>URL for eventing</eventSubURL>
      </service>
      Declarations for other services defined by a UPnP Forum working committee (if any)
        go here
      Declarations for other services added by UPnP vendor (if any) go here
    </serviceList>
    <deviceList>
      Description of embedded devices defined by a UPnP Forum working committee (if any)
        go here
      Description of embedded devices added by UPnP vendor (if any) go here
    </deviceList>
    <presentationURL>URL for presentation</presentationURL>
  </device>
</root>
```

xml	全XMLドキュメントに必要．大文字小文字は区別される		
root	必須項目．xmlnd属性の値としてschemas-UPnP-org: device-1-0をもたなくてはならない．これはUPnP Template Languageを参照してほしい．大文字小文字は区別される．ルートデバイスを記述する他のすべてのエレメント，つまり以下のサブエレメントを含む		
	specVersion		必須項目．以下のサブエレメントを含む
		major	必須項目．UPnPデバイスアーキテクチャのメジャーバージョン．1でなくてはならない
		minor	必須項目．UPnPデバイスアーキテクチャのマイナーバージョン．0でなくてはならない
	URLBase		任意項目．ベースURLを定義する．完全な権限をもつURLを構築する際に用いられる．記述に含まれるすべての関連URLはこのベースURLに添付される． URLBaseが空の場合，あるいは設定されていない場合，ベースURLはデバイス記述の検索元のURLとなる．UPnPベンダにより指定される．単一のURL
	device		必須項目．以下のサブエレメントを含む
		deviceType	必須項目．UPnPデバイスタイプ ※UPnP Forum WCにより定義された標準デバイスの場合は，urn:schemas-UPnP-org: device:から始まり，デバイスタイプの接尾辞，整数のデバイスバージョン(上記リスティング参照)がこれに続かなくてはならない ※UPnPベンダにより定義された非標準デバイスの場合は，urn:から始まり，ベンダが所有するICANNドメイン，:device:，デバイスタイプの接尾辞，コロン，整数のバージョンがこれに続かなくてはならない(domain-name:device: deviceType:v) UPnP Forum WCが定義する，あるいはUPnPベンダが指定するデバイスタイプの接尾辞は，バージョン接尾辞と区切りのコロンを除き，64文字以下でなくてはならない．単一のURI
	friendlyName		必須項目．エンドユーザーのための短い説明．* UPnPベンダにより指定される．64文字未満の文字列
	manufacturer		必須項目．メーカー名．** UPnPベンダにより指定される．64文字未満の文字列
	manufacturerURL		任意項目．メーカーのWebサイト．** ベースURLと関連する場合がある．UPnPベンダにより指定される．単一のURL
	modelDescription		推奨項目．エンドユーザーのための詳細な説明．*．UPnPベンダにより指定される．128文字未満の文字列
	modelName		必須項目．モデル名．** UPnPベンダにより指定される．32文字未満の文字列
	modelNumber		推奨項目．モデル番号．** UPnPベンダにより指定される．32文字未満の文字列
	modelURL		任意項目．モデルのWebサイト．** ベースURLと関連する場合がある．UPnPベンダにより指定される．単一のURL
	serialNumber		推奨項目．シリアル番号．** UPnPベンダにより指定される．64文字未満の文字列
	UDN		必須項目．一意のデバイス名(Unique Device Name)．ルート/組み込みに関わらず，デバイスの一意の普遍的識別子．特定のデバイスインスタンスに対して同じであり続ける必要がある(再起動後も)．デバイスのディスカバリメッセージのNTヘッダの値と一致していなくてはならない． 全ディスカバリメッセージのUSNヘッダの接頭辞と一致していなくてはならない(ディスカバリのセクションで，NTおよびUSNヘッダについて説明している)．uuid:で始まり，それにUPnPベンダが指定したUUID接尾辞が続く．単一のURI
	UPC		任意項目．Universal Product Code．消費者パッケージを識別するための12桁の数字だけのコード．Uniform Code Councilにより管理される．UPnPベンダにより指定される．単一のUPC

〔表10〕デバイスディスクリプション(つづき)

root	iconList			デバイスが一つ以上のアイコンをもつ場合のみ必須項目．UPnPベンダにより指定される．以下のサブエレメントを含む
		icon		推奨項目．コントロールポイントUIでデバイスを描写するアイコン.** 一つのアイコンは以下のいずれかのサイズであることを推奨する(幅×高さ×奥行)：16 × 16 × 1，16 × 16 × 8，32 × 32 × 1，32 × 32 × 8，48 × 48 × 1，48 × 48 × 8．以下のサブエレメントを含む
		mimetype		必須項目．アイコンのMIMEタイプ(RFC 2387)．単一のMIMEイメージタイプ
		width		必須項目．ピクセル単位で表したアイコンの水平方向の寸法．整数
		height		必須項目．ピクセル単位で表したアイコンの垂直方向の寸法．整数
		depth		必須項目．ピクセル毎の色ビット数．整数
		url		必須項目．アイコンイメージへのポインタ(XMLではバイナリデータの直接埋め込みはサポートされていない)．HTTPを介して検索される．ベースURLと関連する場合がある．UPnPベンダにより指定される．単一のURL
	serviceList			必須項目．以下のサブエレメントを含む
		service		必須項目．UPnP Forum WCにより定義された各サービスに対し1度繰り返される．UPnPベンダが追加の標準UPnPサービスによりデバイスを他と区別する場合，各追加サービスに対し1度繰り返される．以下のサブエレメントを含む
			serviceType	必須項目．UPnPサービスタイプ．ハッシュ文字(#，UTF-8の23Hex)を含むことはできない ※ UPnP Forum WCにより定義された標準サービスタイプの場合は，urn:schemas-UPnP-org:service:から始まり，サービスタイプの接尾辞，コロン，整数のサービスバージョンがこれに続かなくてはならない ※UPnPベンダにより定義された非標準サービスタイプの場合は，urn:から始まり，ベンダが所有するICANNドメイン，:service:，サービスタイプの接尾辞，コロン，整数のバージョンがこれに続かなくてはならない(urn:domain-name: service: serviceType:v)UPnP Forum WCが定義する，あるいはUPnPベンダが指定するサービスタイプの接尾辞は，バージョン接尾辞と区切りのコロンを除き，64文字以下でなくてはならない．単一のURI
			serviceId	必須項目．サービスの識別子．このデバイス記述においては一意のものでなくてはならない ※ UPnP WCにより定義された標準サービスの場合は，urn:UPnP-org:serviceId:から始まり，サービスIDの接尾辞がこれに続かなくてはならない(上記リスティング参照)．(この場合，XMLスキーマが各サービスIDに対して定義されていないため，schemas-UPnP-orgの代わりにUPnP-orgが用いられる) ※ UPnPベンダにより定義された非標準サービスの場合は，urn:から始まり，ベンダが所有するICANNドメイン，:serviceId:，サービスIDの接尾辞がこれに続かなくてはならない(urn: domain-name: serviceId: serviceID)UPnP Forum WCが定義する，あるいはUPnPベンダが指定するサービスIDの接尾辞は，64文字以下でなくてはならない．単一のURI
			SCPDURL	必須項目．サービスディスクリプションに対するURL(Service Control Protocol Definition URL)．ベースURLと関連する場合がある．UPnPベンダにより指定される．単一のURL
			controlURL	必須項目．コントロールに対するURL．ベースURLと関連する場合がある．UPnPベンダにより指定される．単一のURL
			eventSubURL	必須項目．イベンティングに対するURL．ベースURLと関連する場合がある．このデバイスにおいては一意のものでなくてはならない．同じイベンティングURLをもつサービスが二つ以上存在することはできない．サービスがイベント変数をもたない場合は，サービスにイベンティングはない(イベンティングのセクション参照)．サービスにイベンティングがない場合，このエレメントは存在するが，空となる(つまり<eventSubURL></eventSubURL>となる)．UPnPベンダにより指定される．単一のURL
	deviceList			ルートデバイスが組み込みデバイスをもつ場合のみ必須項目．以下のサブエレメントを含む
		device		必須項目．UPnP Forum WCにより定義された組み込みデバイスに対し1度繰り返される．UPnPベンダが標準UPnPサービスを組み込むことによってデバイスを他と区別する場合，各組み込みデバイスに対し1度繰り返される．上に定義されているとおり，ルートサブエレメントデバイスに対しサブエレメントを含む．入れ子の構成になる
	presentationURL			推奨項目．デバイスのプレゼンテーションのためのURL．ベースURLと関連する場合がある．UPnPベンダにより指定される．単一のURL

*：ローカライズされていることが望ましいとされる(ACCEPT-/CONTENT-LANGUAGEヘッダ).
**：ローカライズすることが可能(ACCEPT-/CONTENT-LANGUAGEヘッダ).

〔表11〕サービスディスクリプション

```
<?xml version="1.0"?>
<scpd xmlns="urn:schemas-upnp-org:service-1-0">
  <specVersion>
    <major>1</major>
    <minor>0</minor>
  </specVersion>
  <actionList>
    <action>
      <name>actionName</name>
      <argumentList>
        <argument>
          <name>formalParameterName</name>
          <direction>in xor out</direction>
          <retval />
          <relatedStateVariable>stateVariableName</relatedStateVariable>
        </argument>
       Declarations for other arguments defined by UPnP Forum working committee (if any)
          go here
      </argumentList>
    </action>
    Declarations for other actions defined by UPnP Forum working committee (if any)
      go here
    Declarations for other actions added by UPnP vendor (if any) go here
  </actionList>
  <serviceStateTable>
    <stateVariable sendEvents="yes">
      <name>variableName</name>
      <dataType>variable data type</dataType>
      <defaultValue>default value</defaultValue>
      <allowedValueList>
        <allowedValue>enumerated value</allowedValue>
        Other allowed values defined by UPnP Forum working committee (if any) go here
      </allowedValueList>
    </stateVariable>
    <stateVariable sendEvents="yes">
      <name>variableName</name>
      <dataType>variable data type</dataType>
      <defaultValue>default value</defaultValue>
      <allowedValueRange>
        <minimum>minimum value</minimum>
        <maximum>maximum value</maximum>
        <step>increment value</step>
      </allowedValueRange>
    </stateVariable>
    Declarations for other state variables defined by UPnP Forum working committee
      (if any) go here
    Declarations for other state variables added by UPnP vendor (if any) go here
  </serviceStateTable>
</scpd>
```

<table>
<tr><td>xml</td><td colspan="4">全XMLドキュメントに必須．大文字小文字は区別される</td></tr>
<tr><td rowspan="8">scpd</td><td colspan="4">必須項目．xmlns属性の値としてurn:schemas-UPnP-org: service-1-0をもたなくてはならない．これはUPnP Template Languageを参照してほしい．大文字小文字は区別される．サービスを記述する他のすべてのエレメント，つまり以下のサブエレメントを含む</td></tr>
<tr><td rowspan="3">specVersion</td><td colspan="3">必須項目．以下のサブエレメントを含む</td></tr>
<tr><td>major</td><td colspan="2">必須項目．UPnPデバイスアーキテクチャのメジャーバージョン．1でなくてはならない</td></tr>
<tr><td>minor</td><td colspan="2">必須項目．UPnPデバイスアーキテクチャのマイナーバージョン．0でなくてはならない</td></tr>
<tr><td rowspan="4">actionList</td><td colspan="3">サービスにアクションがある場合のみ必須項目(各サービスにはゼロ以上のアクションがある)．以下のサブエレメントを含む</td></tr>
<tr><td rowspan="3">action</td><td colspan="2">必須項目．UPnP Forum WCにより定義された各アクションに対し1度繰り返される．UPnPベンダがアクションを追加することでサービスを他のデバイスと差別化する場合，各追加アクションに対し1度繰り返される．以下のサブエレメントを含む</td></tr>
<tr><td>name</td><td>必須項目．アクション名．ハイフン文字(-，UTF-8の2D Hex)，ハッシュ文字(#，UTF-8の23 Hex)を含むことはできない
※ UPnP Forum WCにより定義された標準アクションの場合は，X_あるいはA_から始まってはならない
※ UPnPベンダにより指定され，標準サービスに追加された非標準アクションの場合，必ずX_から始まる．32文字未満の文字列</td></tr>
<tr><td>argumentList</td><td>アクションにパラメータが定義されている場合のみ必須項目(各アクションにはゼロ以上のパラメータがある)．以下のサブエレメントを含む</td></tr>
</table>

〔表11〕サービスディスクリプション(つづき)

scpd	actionList	action	argumentList	argument	必須項目．各パラメータに対し1度繰り返し．以下のサブエレメントを含む
				name	必須項目．正式パラメータの名前．アクションが起こすエフェクトをモデル化するステートバリアブル(状態変数)の名前．ハイフン文字(-，UTF-8の2D Hex)を含むことはできない．32文字未満の文字列
				direction	必須項目．引き数が入力パラメータか出力パラメータかを示す．inまたはoutでなくてはならない．in引き数は，必ずout引き数より先になる
				retval	任意項目．返り値として0または一つの引き数を識別する．retvalが含まれる場合，必ず最初のout引き数となる(エレメントのみ．値なし)
				relatedStateVariable	必須項目．状態変数の名前でなくてはならない
	serviceStateTable	必須項目(各サービスには必ず1以上の状態変数がある)．以下のサブエレメントを含む			
		stateVariable	必須項目．UPnP Forum WCにより定義された各状態変数に対し1度繰り返される．UPnPベンダが状態変数を追加することでサービスを他のデバイスと差別化する場合，各追加変数に対し1度繰り返される．sendEvents属性は，この状態変数の値が変化したときにイベントメッセージが生成されるかどうかを定義する．イベント化されていない状態変数ではsendEvents="no"となり，デフォルトはsendEvents="yes"．以下のサブエレメントを含む		
			name	必須項目．状態変数の名前．ハイフン文字(-，UTF-8の2D Hex)を含むことはできない ※UPnP Forum WCにより定義された標準変数の場合は，X_あるいはA_から始まってはならない ※UPnPベンダにより指定され，標準サービスに追加された非標準変数の場合，必ずX_から始まる 32文字未満の文字列	
			dataType	必須項目．XML Schema，Part 2: Datatypesで定義されたデータ型と同じ．標準状態変数に関してはUPnP Forum WCにより定義され，拡張子に関してはUPnPベンダにより設定される．以下のいずれかの値でなくてはならない	
				ui1	符号なし1バイト整数型
				ui2	符号なし2バイト整数型
				ui4	符号なし4バイト整数型
				i1	1バイト整数型
				i2	2バイト整数型
				i4	4バイト整数型．−2147483648と2147483647の間でなくてはならない
				int	固定小数点型，整数値．最初に符号がある場合がある．また，最初にゼロがある場合がある(通貨記号なし，少数の左に桁のグルーピングはない．例：カンマはない)
				r4	4バイト浮動小数点型．floatと同じフォーマット．3.40282347E+38と1.17549435E − 38の間でなくてはならない
				r8	8バイト浮動小数点型．floatと同じフォーマット．負の値の場合 −1.79769313486232E308と −4.94065645841247E-324の間，正の値の場合4.94065645841247E-324と1.79769313486232E308の間〔つまりIEEE 64ビット(8バイト)の倍〕になる
				number	r8と同様
				fixed.14.4	r8と同様だが小数点の左14桁まで，右4桁までとなる
				float	浮動小数点型．仮数部(小数の左)および/またはべき指数には符号を付けることができる．仮数部および/またはべき指数には最初にゼロをつけることもできる．仮数部の少数文字はピリオド．つまり，仮数部の全体は分数桁からピリオドで分けられている．仮数部はべき指数からEで分けられている(通貨記号なし，仮数部に桁のグルーピングはない．例：カンマはない)
				char	Unicode文字列．長さ1文字
				string	Unicode文字列．長さ制限はない
				date	時間データのない，ISO 8601フォーマットのサブセットの日付

〔表11〕サービスディスクリプション(つづき)

scpd	serviceStateTable	stateVariable	dataType	dateTime	任意の時間はあるがタイムゾーンのない，ISO8601フォーマットの日付
				dateTime.tz	任意の時間および任意のタイムゾーンをもつ，ISO8601フォーマットの日付
				time	日付もタイムゾーンもない，ISO 8601フォーマットのサブセットの時間
				time.tz	任意のタイムゾーンはあるが日付のない，ISO 8601フォーマットのサブセットの時間
				boolean	0，false，またはfalseを意味するno．1，true，またはtrueを意味するyes
				bin.base64	MIMEスタイル，Base64エンコードされたバイナリBLOB．3バイトを要し，それを四つのパートに分け，各6ビットピースをオクテットにマッピング(3オクテットは4としてエンコードされる)．サイズ制限はない
				bin.hex	オクテットを表す16進数．各ニブルを16進数として扱い，個別のバイトとしてエンコードする(1オクテットは2としてエンコードされる)．サイズ制限はない
				uri	Uniform Resource Identifier
				uuid	Universally Unique ID(普遍的な一意のID)．オクテットを表す16進数．任意の埋め込みハイフンは無視される
			defaultValue	推奨項目．初期値．UPnP Forum WCにより定義，あるいはUPnPベンダにより指定．データ型と一致しなくてはならない．allowedValueListまたはallowedValueRange制数を満たしている必要がある	
			allowed ValueList	推奨項目．適切な文字列値を列挙する．文字列以外のデータ型に対しては禁止されている．allowedValueRange，またはallowedValueListのどちらかを指定できる．サブエレメントの順序は決められている(例：NEXT_STRING_BOUNDED参照)．以下のサブエレメントを含む	
				allowedValue	必須項目．文字列変数に対する適切値．標準状態変数に関してはUPnP Forum WCにより定義され，拡張子に関してはUPnPベンダにより設定される．32文字未満の文字列
			allowed ValueRange	推奨項目．適切な数値の境界を定義し，数値に対するレゾリューションを定義する．数値データ型の場合のみ定義される．allowedValueRange，またはallowedValueListのどちらかを指定できる．以下のサブエレメントを含む	
				minimum	必須項目．包括的な下限．UPnP Forum WCにより定義，あるいはUPnPベンダにより指定．単一の数値
				maximum	必須項目．包括的な上限．UPnP Forum WCにより定義，あるいはUPnPベンダにより指定．単一の数値
				step	推奨項目．インクリメント演算子のサイズ，つまり演算子vのsの値 $= v + s$．UPnP Forum WCにより定義あるいはUPnPベンダにより指定．単一の数値

〔表12〕ベンダ拡張

拡張タイプ	標準	非標準
デバイスタイプ	urn:schemas-UPnP-org: device:deviceType: v	urn:domain-name: device:deviceType: v
サービスタイプ	urn:schemas-UPnP-org: service:serviceType: v	urn:domain-name: service:serviceType: v
サービスID	urn:UPnP-org: serviceId:serviceID	urn:domain-name: serviceId:serviceID
アクション名	X_またはA_で始まらない	X_で始まる
状態変数名	X_またはA_で始まらない	X_で始まる
デバイスまたはサービス記述のXMLエレメント	UPnP Template Languageで定義	XMLネームスペースにより範囲が設定され，X_で始まるエレメント内にネスト化された任意のXML
デバイスまたはサービス記述のXML属性	UPnP Template Languageで定義	XMLネームスペースにより範囲設定され，X_で始まる任意の属性

プログラミング入門シリーズ 好評発売中
Visual Basicでわかる物理
物理とプログラミングの双方向理解を深める
山田 盛夫 著
B5変型判 240ページ+口絵4ページ CD-ROM付き
定価2,940円(税込) ISBN4-7898-3703-3
CQ出版社 〒170-8461 東京都豊島区巣鴨1-14-2 販売部 TEL.03-5395-2141 振替 00100-7-10665

IPパケットの隙間から

嘘と呪いの後で

56

祐安重夫

4月号でエープリルフール，5月号でそのようなものを書いた呪いと話を進めてきたが，この原稿を書いている現実世界ではまだかろうじて3月である．しかし，現実世界は恐い．何しろ本当に戦争が起きてしまったのだから．

まあ，戦争についての議論は，とりあえず『アイアンマウンテン報告』にでもまかせておこう．この天下の偽書は，

```
http://cruel.org/books/ironmountain.pdf
```

で読むことができる．

さて，先月話題にしたサンフランシスコのMさんはといえば，3月の半ばをすぎてから音沙汰がない．最後のメールでは「税金の季節で忙しい．女房が東海岸に旅行に行っているので，自由でのんびりできるかと思ったら大間違い．忙しくてLinuxどころではなかった」と書いてきた．どうやらアメリカの戦費調達に「協力」しているようだ．

前回はメーラとしてMewをインストールさせる気にまではさせたが，何とMさんはLinuxのインストール時に開発環境を入れていなかったことが判明した．その後，オンラインでのパッケージのアップデートをいくつかしたそうで，追加で開発環境を入れようとしたら，パッケージの依存関係に問題が出てきてインストールできないという．

もっとも，Mさんの場合，コンピュータに限らず，ドアの鍵からエアコンのリモコンまで，自分のものだけではなく他人のものまで信じられない方法で壊してしまう人なので，実際にそれを再現できるかどうかは不明である．

さらにMさんにとっては不幸だが，こちらにとっては幸いなことに，MさんがノートPCのBIOS設定を変更したら，Linuxのデスクトップが正常に動作しなくなった．Mさんのことだから，BIOSに異常な設定をした可能性はかなり高いが，本人はこういうことには慣れているので，これまでの作業内容や設定ファイルはちゃんとメモやバックアップしてあった．ということで，今度はフルインストールでLinuxを再インストールすることになった．怪我の巧名というか，祟りや呪いというべきか，これで今後の作業を含めて，一応安心できる環境が整った．

MewはEmacs用のアプリケーションだが，動作上いくつかのCで書かれたコマンドをコンパイルしてインストールする必要がある．これらのコマンドのコンパイル済みのバイナリを，メールに添付して送ってしまえば解決できそうに思えるが，実際には開発環境がないと最初のconfigureができない．メールと電話で，そこまでのサポートするのはかなり面倒だ．

しかし，Mさんは新規インストールの後，3日間をかけて自分だけで再インストール寸前のEmacsまで動作していた環境を復元してしまった．まったく，ソフトウェアを含むコンピュータが壊れることに慣れている人はさすがである．そして今度は，Mewについても，最新版を自力でインストールしてしまった．それでも動作しないというメールがあったが，原因はload-pathを設定していなかったためで，すぐに実際にMewでメールを書いてくるようになった．

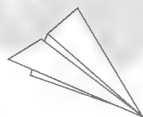
ところで，ふと気がつくと，筆者が使っているMewのバージョンは1.93である．おもにBSD/OS上でメールを読み書きしていたので，そこにインストールされていたEmacs 19.28/Mule 2.3(末摘花)ではこれ以上のバージョンのMewが動作しなかったのだ．現在この原稿を書いているのも，そのMuleである．

とりあえず，Linuxでemacs-20.7とMew 3.2という組み合わせでテストはしてあったが，Linux上で日常的に使っているxemacs-21.1.8では，なぜかこのバージョンのMewは動作しない．そこでxemacsの21.4.12をインストールし，メールの読み書きもLinux環境に移行することにした．こうしておかないと，Mさんのサポートもできない．

その後，ispellでのスペルチェックで，単語登録をするのはどうするのかとか，細かい質問はいくつかあったが，かなり大物の質問がメールで届いた．Unicodeで繁体字中国語と英語と日本語の混在したメールは，どうやったら読めるかというものである．答は簡単で，Mule-UCSとintlfontsをインストールすればいい．インストール方法の詳細をメールで送ったら，「げっ．なんでもできるのだなぁ．もうあきらめていた．驚いたね」というメールが返ってきた．どうやら，できないと思い込んでいたようだ．

ついでに，うちでもそれらをインストールしたところ，これまでUTF-7エンコーディングされた日本語という，ふざけたエラーメッセージを返してきていた一部のメールサーバからのエラーメールが読めるようになった．おかげでそのメールサーバのユーザーが存在しないことが判明し，安心してメーリングリストから削除できるようになった．それから中国語のSPAMメールが正常に表示できるようになったが，こちらは表示内容に関係なく判読できないのだから無意味である．

これから出てくるだろう質問を予測して，w3mとemacs-w3mのインストールと，wvとxlhtml，ppthtmlをインストールして互換性のない勝手なフォーマットの添付ファイルをMewとemacsだけで表示する方法などについても解答を用意してあるのだが，どうやらUnicode対応の前に税金の季節になってしまったようで，その後の展開はまだない．

すけやす・しげお　インターメディアアクセス

● オーディオブームの時代

1970年ごろ，オーディオの全盛時代があった．少しでもいい音を，あるいは少しでも自分にあった音を聞こうと，アンプやスピーカ，プレーヤなどを，メーカーやグレードにこだわらずいろいろと買いそろえ，組み合わせて自分なりのシステムを作った．これをコンポーネントステレオと呼んだ．

たとえば，電源トランスは低インピーダンスで高性能なものは重かったから，20kgぐらいあるアンプも少なくなかった．秋葉原で購入した大きくて重いダンボールの箱をやっとの思いで駅まで引きずっていったものだ．プレーヤも慣性モーメントの大きなもののほうが安定した回転ができる．したがって，高級品は大きく重いターンテーブルを採用しているものが多く，当然ながら大きくて重かった．高級なオープンリールのテープデッキにいたっては，10号(10インチ)リールを二つ悠々と前面パネルの上で回転させることができる，大きな筐体が必要だった．このようにコンポーネントステレオは，高性能のものは大きく重いというのが常識だった．

各メーカーは，新製品開発にとくに力を入れ，アンプやプレーヤ，スピーカなどで新しい技術が次々と投入されていたので，頻繁に買い換えた．そんな具合だから，巨大なアンプやデッキなどをそれぞれ2個ぐらいもっているのはオーディオマニアとしては普通だった．

こういったコンポーネントの数々を購入すると，決まって付いてくるのが大きなダンボールと発泡スチロールだ．入っているものが大きく重いから，段ボールも並のものではない．子供なら中に入ってしまえるくらい大きく，厚さも通常の2倍くらいあった．

そんな段ボールや，発泡スチロールの処理にはたいへん困った．新しいコンポーネントの一部を買い換えると，箱に入れて押入れにしまう必要があった．また，中古販売の業者に売ってしまう場合にもダンボールはとっておく必要があった．引越しのときは，段ボールは繊細なオーディオコンポにとって必須だった．当時の引越し業者は，今のようにサービス業だとは一つも思っていなかったので，業者にまかせると必ず傷だらけになって後悔するのが関の山だったからだ．

そんなこんなで，押入れの中は，なかなか捨てることのできないダンボールと発泡スチロールで一杯になった．

● パソコンの外箱

あれから30年，オーディオブームはパソコンブームに変わった．変わらないのは，いまでもコンピュータは，発泡スチロールと大きな段ボールで梱包されていることだ．最近は環境への配慮から発泡スチロールは減り，ダンボールを折った梱包材などが登場しているが，本体やディスプレイなどの大きなものには，まだまだ発泡スチロールが使われている．発泡スチロールが使われていない場合は，その分だけ段ボールがやたらと大きくなった気がするし，必要ないものまで段ボールに入るようになった．たとえば，ルータ，ハブ，スキャナ，プリンタなどだ．

最近は，自宅でも複数のパソコンを保有する例が増えてきているため，ダンボールが山のように発生する．

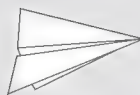
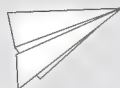
引越しはもとより，オークションに出したりすることもある．パソコンショップでも，新製品が売れなくなった分，中古品の販売に力を入れている．中古品はダンボールや付属品がきちんと残っているほうが高く売れるので，箱を保存している人も多いのだろう．

結局，段ボールが山のようにたまっている状況は，今も昔も変わらないのではないだろうか．日本人の狭い家を占拠してしまう段ボールの山には，拡販一途のメーカーにも責任の一端がある気がしてならない．

最近は一家に複数のパソコンがあるのも珍しくなくなった．多くのパソコンの箱を保存している家庭も多いかもしれない．ある賃貸マンションに住んでいる友人は，転勤のために箱はほとんど残してあるという．そして西日の当たる部屋に積み上げてあるので，その部屋はカーテンの開け閉めすらできないそうだ．読者も同じような悩みはないだろうか．

● フルカラーの段ボール

最近は，ダンボールの外側にフルカラーで製品の概略図や仕様が印刷されていることも多い．箱を手にとって眺めるだけでいろいろなことがわかるので，たいへんありがたい．

しかし，大きな段ボールの箱に入ったメーカー製のプリンタやPC本体まで，外箱にきれいにカラーで印刷されていることも多くなった．製品寿命が短いので，店頭で箱ごと積み上げ，ディスプレイすることも多くなったからだろうか．それに，型番と製品名だけ書いてあるより，製品のカラー写真がついていたほ

うが在庫整理もしやすいのかもしれない．しかし，ここまで外箱をきれいにしなくてもよいとは思うのだが．

そうやって，店に並んでいるパソコンや段ボールを眺めていて，ふと気が付いた．パソコンやプリンタはどこのメーカーも似たような大きさで，形も似ていることが多い．薄型のデスクトップ型，中型のデスクトップ型，それからノートも，B5サイズとA4サイズがほとんどで，厚さもそんなに大きく差のあるものはない．いっそのこと，メーカーを超えてダンボールの外箱を規格化してもよいのではないだろうか．3，4種類の外箱規格を作れば，メーカーにこだわらず，みな使える気がする．実際，デスクトップのパソコンケースの段ボール箱は，一部規格化されているはずだ．

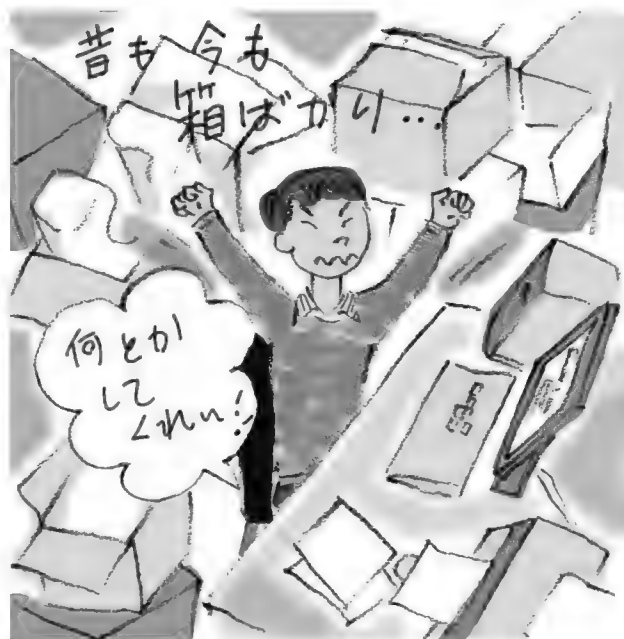
● 段ボールと発泡スチロールの規格化

そうなれば，家庭では段ボールをとっておかなくてもよくなる．とっておくとしても，最少限の個数でよい．必要なとき，箱だけ購入すればいい．いままで埋まっていた押入れや，タンスの上のスペースが大きく開くことになる．

定形の箱になるから流通コストも削減できるはずだ．当然，資源の再利用も可能だし，環境に優しくなる．ゴミ問題やリサイクル問題にも，かなりの効果がある気がする．

外箱はたぶん，必要なときにパソコンショップや引越し業者から買えばいい．中古販売のオークションでも箱を用意してくれる可能性だってある．逆に，自分のパソコンの箱を引き取ってもらうといったことも可能かもしれない．

外観で見分けがつきづらくなる分，製品管理上の問題が出てくるかもしれない．ダンボールの外箱にカラーできれいに印刷しているのは，パソコンとプリンタくらいのような気がする．それくらいは運用でカバーできるはずだ．シールで型番やバーコードを貼り付ければいい．実際，ほかの電気製品などでは製品名と型番しか書いてないものも多いではないか．

日本でもっとも売れている家電はパソコンだ．国内だけでも年間1300万台のパソコン，700万台のプリンタが現在も売れ続けている．逆に考えれば，これだけの数の段ボールと発泡スチロールが出ていることになる．これでは，まさに売りっぱなしではないか．パソコン関連のダンボールを規格化するだけで，ゴミ問題やリサイクル問題，へたをすると住宅問題にまで効果が

出たりするかもしれない．メーカーも，そろそろ対処法を考えてもよいのではないかと思う．

そうは思うのだが，利益率が少なく，そんな余裕はないというのも実際のところだろう．自由競争の民間企業主導では，この種の試みは成功しないかもしれない．こういうときこそ，国がニラミを利かせてほしいものなのだが……．

考えたのだが，発泡スチロールばかりはどうしようもない．製品によって形が違うため，規格化はとても難しい．圧縮して保存することもできないし．燃やせばダイオキシンをはじめとする有害な物質を出す．

何か良い手立てはないだろうかと，最近考えている．低コストで実現でき，場所を取らずに保存できる，そんないい案を考えついたら，自由に梱包できるので「自由梱包」とでも名づけ，発明大賞にでも応募しようと考え，毎日真剣に頭を悩ませている．

あさひ・しょうすけ　テクニカルライター
イラスト　森　祐子

Engineering Life in

専門分野の第一線で活躍するエンジニア

▍今回のゲストのプロフィール
加藤比呂武(かとう・ひろむ):甲南大学理学部および経営学部卒業,日本ディジタルイクイップメント(株)ではDEC Rdbを使用した大規模データベースプロジェクトの設計,開発を行う.その後,1995年に日本オラクル(株),1996年より米国オラクル社で,TPC(後述)を中心としたOracle8,Oracle8iデータベースの性能評価やチューニングに従事する.1999年からは,米BroadVision,Inc.社に勤める.現在,J2EEなどを使用したインターネットソフトウェア製品BroadVision7の開発とチューニングを行う.著書には共著で『Oracle8i最新テクノロジガイド』(アスキー出版)などがある.趣味はハイキング,水泳,コンサートに行くこと.2002年の最高の思い出は,ヨセミテ国立公園のハーフドームハイキングコース(往復10時間)を歩いたこと.

☆ 性能評価の分野を追いかけ続けてアメリカに

トニー かつてはDECにいらっしゃったのですよね? 個人的には学生の頃からお世話になった会社です.大学のコンピュータがIBMとDECで統一されていました.アセンブラのクラスではPDP-11/10を使いました.

加藤 あ～懐かしいですね.私はPDP-11/34とかでした.

トニー ほかのクラスでは,回路シミュレータのSPICEやFORTRAN系などではIBMのメインフレームを使ってました.その後,最初の職場ではDECの主要OSであるVAX/VMSで社内が統一されていて社内メールなどを行っていました.エンジニアリングのほうは,今はなくなったApolloを使っていました.

加藤 Tonyさん,けっこう占いですね(笑).私はずっとDECではVMS上で走るRDB(リレーショナルデータベース)のアプリケーションの開発を日本でやっていました.当時は,DECがIBMを抜くんだ!という勢いがまだまだあり,技術で引っ張っていく会社だというイメージがありました.つまり,自分達の会社がコンピューティングの流れを作っていくという自負が非常に強く,技術力を高く評価する会社でした.ですから,エンジニアにとっては非常に良い会社でした.とくにデータベースの分野では,著名な技術者・エンジニアをたくさん抱えていました.たとえば,ACM Turing賞[注1]を取られたJames Gray氏[注2]がいた頃でした.当時は,STDL[注3]を採用したプロジェクトをやっており,そのプロジェクトやTPC[注4]関連のことでDECの本社のほうによく出かけました.そのときにGray氏とも仕事をする機会がありました.

トニー DECからオラクルに入られたときは戸惑いなどはありませんでしたか?

加藤 結局,私の関連していたグループはオラクルの一部になり,オラクルの社員になりました.大きな違いはたくさんありました…….まず,オラクルがソフトウェアだけの会社だし,オープンアーキテクチャの概念で作られたところがありますよね.DECは,最後はVMSに賭けて駄目になっちゃったし,会社が大きくてなかなか方向転換するのに時間がかかったと思います.オラクルではいろいろなプロジェクトがあったり,営業やマーケットドリブンの部分が多い印象がしました.全体の雰囲気はすごくアグレッシブで,「イケイケ的」なところがあると思います.

トニー オラクルで実際にシリコンバレーのほうに来られたのですよね? あのRedwood Shoresの筒の中[注5]ですよね?

加藤 そうです.オラクルの一部になってから,1年ほど日本にいました.そして,もう少しパフォーマンスチューニングの分野を追求したいと思い,渡米しました.

☆ 仕事と職場について

トニー 現在のお仕事について教えてください.

加藤 おもな仕事は,BroadVision, Inc.のソフトウェア製品のパフォーマンス解析・向上です.たとえば,CやC++であればquantify,Javaならjprobeなどを使ってプロファイリングを行います.

また,Java VMのヒープサイズを調べ,オブジェクトの作成と消滅の度合いを調べたり,CPUのカーネルやユーザーの動作,ディスクI/Oの割合を見たり,転送レートや負荷を上げていくにつれてアプリケーションの性能がスケールしていくかとか,解析によって性能アップにつながる方法を見つけていきます.まあ,現在の職場がかなりこじんまりして,スタートアップにも似た雰囲気をもっており,ほかの仕事もたくさんやらされますが(笑).

トニー それはどういうことですか?

加藤 パフォーマンス解析の仕事は社内ではごく小人数なのですが,そのほかにデータベースのクエリプランなどを調べ,クエリが最適化されているかどうかを評価し,対策を考えたり,パートナー企業のベンチマークを手伝ったりしています.

トニー 小さな会社に入られてどうですか?

加藤 社長らが台湾人で,開発グループもアジア系の人が多いのでなかなか居心地がよく,フレンドリーな感じです.白人系の人は営業やマーケティングの人がほとんどですね.小規模の会社なので,土日とか夜とか関係なしに「困ったことがあるから助けて」という内容のメールがドンドン入ってきます.

注1:1966年に始まったACM(計算機学会)のチューリング賞は,ノーベル賞や数学界におけるフィールズ賞にも匹敵する世界最高権威の賞であり,情報科学の基礎分野の発展に多大な貢献をした研究者に毎年贈られる.その受賞者にはアルゴリズムの分野の研究者が多い.

注2:同氏は,現在Microsoft社で研究を続けている.

注3:Structured Transaction Definition Language

注4:TPCベンチマーク:Transaction Processing Performance Council(TPC)が提供するベンチマークであり,データベースシステムのパフォーマンス測定における業界標準.現在,各データベースベンダーによって実施されているベンチマークテストとしては,TPC-C,TPC-H,TPC-R,TPC-Wなどがある.

注5:オラクルの本社ビルは,独特な形をした筒形のビルで,高速道路101から見える.

Silicon Valley

H. Tony Chin

対談編

トニー シリコンバレーによくあるパターンですよね．

加藤 ただ大幅なレイオフとかあったりして，その後もまた小出しにレイオフがあるので，誰が次やられるかわからない状態が続き，あまりよくないですね．会社の業績が良いときはみんな協力的な仕事の進め方なんですが，レイオフが始まると自分の身の危険を感じて，自分の職を守るほうにエネルギーが動くので，精神的にやっぱり疲れます．

☆ 自分の仕事の結果を見えるようにする

加藤 また，私の場合は一人でやっている仕事だし，変わった内容の仕事をしているので，自分のやっていることを表面に出す方法には常に気をつけています．普通のエンジニアだと，コードを書いて期日に出荷するという明確な尺度があります．私の場合，細かい仕事のテーマや期限も自分で決めて進めるので，どれだけ仕事したかをうまくアピールすることが重要です．

トニー 上司の方もとくに指示しないのですか？

加藤 具体的な指示というより，方向性は示してくれます．かなり専門的な分野なので，実際現場で直に仕事をしていないと具体的な指示が出せないからです．まず，自分で方向性とかプランを立てて，上司と相談してから指示するという流れになります．自分で自分の指示を出しているのとほぼ同じです．

それでなおかつ明確な成果物がないのですから，自分なりに結果を出さなければなりません．ですから，まわりに自分のやっていることが見えるように，仕事のレポートや結果をイントラネットで社内に公開して履歴を見えるようにしています．あとは，社内で技術者向けの発表会"Tech Seminar"が定期的にあるのですが，それで成果を積極的に発表しています．

トニー 大体，どの会社でも社内発表会みたいなものがありますよね．何となく純粋な研究に近いお仕事みたいですよね，自分で仕事の道筋を決めるところは．

加藤 そうですね，仕事としては楽しいですよ．でも成果を見せるところではかなりプレッシャーがあります．

☆ 自分の知識をアップデートして行く努力

トニー インターネットやデータ管理の分野は，ソフトウェア業界ではもっとも大きな分野なのですが，動きは早いですよね？

加藤 まあ，ハイテク全体が生き物みたいなので，むろんこの分野も動きは早いです．データベースの部分は成熟している分野で，もともとリレーショナルモデルは，数学の集合理論をベースに学術的に発展してきたところもあります．

しかし，現在ではデータベースというものが，アプリケーションサーバやエンタープライズサーバの一部としてとらえられています．つまり，以前はデータベースそのもののチューニングに専念していればOKでしたが，その先にあるインターネットやアプリケーションレイヤが発展していて，それがボトルネックになることも多く，これらをチューニングすることが非常に重要になってきています．また，ハードウェアリソース，ストレージ系も新しい動きがドンドンあるし……．

加藤比呂武氏

しかしその一方で，アプリケーションサーバを使った標準的なベンチマークが認知されるのはまだまだ時間がかかるし，その間にJavaやXMLの仕様が変わったりしています．だから，勉強したり，新しい動向を把握しておくことがたいせつになります．

トニー 技術者・エンジニアとして勉強と新しい動向を把握するのは必須ですよね．私も大学生の頃，教授が「これから教えることは，君達が卒業する頃には使いものにならないと思うから，エッセンスと勉強の仕方をしっかり覚えること」と言っていたのを覚えています．それで，具体的にどういう方法で自分の知識をアップデートしていますか？

加藤 勉強の仕方は同感ですね．私の場合は，二つに分けて考えています．コアの知識として自分が常に追いかけていきたいものと，トレンドとして見ていきたいもの．後者は，社内の人や知人と情報交換したりしてテーマを拾っていきます．

あとは言語系とかになると，勉強をする時間をちゃんと作ってやらないと駄目ですよね．とにかく，やらないといけないことが多いので，やり方に工夫が必要だと思います．たとえば，新しい言語などの場合は，その言語のエッセンスのところだけをしっかり見るとか……細かい部分は実際仕事で使う場面でまた細かくみていけばよいわけですから．

トニー つまり，コアの部分もトレンドの部分も優先順位をつけるために工夫が必要なわけですね．ネットワーキングを使ったりしてどれが自分にとってたいせつかを見つけていく……．

加藤 そうですね，自分の市場価値をしっかりつかんでおかないと，レイオフとか新しいチャンスがあったときにどう動いてよいかわからなくなりますから．

対談を終えて：

エンタープライズ系のソフトでもかなり中核で専門的な仕事やっておられるため，技術的な話題が豊富な対談だった．加藤氏は，日本におられた時代からずっと技術の第一線で活躍されており，その後も管理方面にいかず，技術職で続けているのは相当な技術力があるからだと思う．今後は，日本にも自分の得たものを還元していきたいのだとか．アメリカだけでなく，今後発展していく中国のソフト業界など，グローバルに仕事を進めていきたいそうだ．

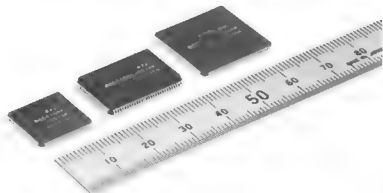
トニー・チン htchin@attglobal.net WinHawk Consulting

HARD WARE

●16ビットマイコン

M32C/81グループ

- M16Cとして最高速度の40MHz動作(最小命令実行時間25ns),1.65μsのA-D変換速度を達成.
- インテリジェントI/O,CAN通信機能など豊富な周辺機能を内蔵し,搭載機器の高機能化を実現.
- −40℃〜+125℃の広温度保証品もサポートしており,車載電装品の高機能化を実現.
- 従来品と命令,周辺機能,ピン配置の互換性を維持しているため,既存のマイコンを置き換えるだけで,搭載機器の処理能力を大幅に向上.
- 128KバイトROMおよび10K/12KバイトRAMの内蔵メモリを搭載.

■ **三菱電機(株)**
サンプル価格:¥1,500
TEL:03-3218-9450

●16ビットマイコン

MB90330シリーズ

- USB通信の簡易ホスト機能をもったUSBマクロを搭載しているため,ディジタルAV機器やパソコン周辺機器間で,パソコンを介さずにダイレクトな通信が可能.
- 最大24KバイトのRAMと,用途に応じてどちらかを選択可能な,最大384Kバイトのフラッシュメモリまたは最大256KバイトのROMを搭載するため,大量のプログラムやデータの格納が可能.
- プロセステクノロジは,0.35μmのCMOSを採用.
- 最小命令実行時間は,41.6ns/6MHz.
- 8/10ビット,16チャネルのA-Dコンバータを搭載.
- クロック同期式シリアルに対応したSIOおよび調歩同期式のUARTを内蔵.
- 電源電圧は,3.0〜3.6V.
- 動作温度は,−45℃〜+85℃(USB使用時は,0℃〜70℃).

■ **富士通(株)**
サンプル価格:¥900〜¥2,400
TEL:042-532-1397
E-mail:edevice@fujitsu.co.jp

●DVDディジタルバックエンドプロセッサ

TMS320DVD20

- DVD/HDDレコーダ向け製品として,映像と音声を圧縮/伸張できる機能を備え,1チップで録画や再生,同時録画再生を実現.
- 同社が新たに開発した,再生やスロー再生時に,等倍速と変わらない滑らかな映像と自然な音声の再生が可能.
- DSP「TMS320C55x」(160MHz),32ビットRISCプロセッサ「ARM9E-S」(108MHz),ビデオコーデック(MPEG-1,MPEG-2,DV),ストリームコントローラおよびOSD用グラフィックエンジンを集積.
- SDRAMコントローラにより,外付けのSDRAMを1個に統合.
- 0.13μmの6層銅配線プロセスを採用.
- リファレンスソフトウェアとAPIにより,開発期間の短縮を実現.

■ **日本テキサス・インスツルメンツ(株)**
サンプル価格:〜$30(100,000個時)
FAX:0120-81-0036
URL:http://www.tij.co.jp/pic/

●ブロードバンドネットワークプロセッサ

CX8611xシリーズ

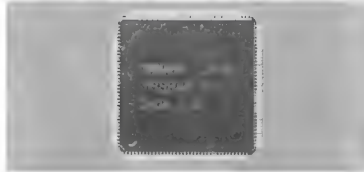
- Ethernet,USB,HomePNA,Home Plug,IEEE802.11a/b/g,およびその他のLANメディアに対応する複数のパソコン,ネットワーク機器に高速WANインターネットアクセスを提供.
- WAN側アクセス方式としては,DSL,ケーブル,Ethernetを含むブロードバンド技術をサポート.
- ARM926EJ-Sプロセッサをコアとし,200MHz/300MHzの2種類のCPUクロック品を用意.
- メモリマネージメントユニットの搭載により,Linux,VxWorks,Windows CE.NETなどの主要OSをサポート.
- VPNアプリケーション対応の高性能暗号エンジンを搭載.
- USBホスト機能をもつため,プリンタサーバ機能およびUSB接続のブロードバンドゲートウェイを実現可能.

■ **コネクサント・システムズ(株)**
価格:$15(10,000個時)
TEL:03-5371-1520 FAX:03-5371-1501
URL:http://www.conexant.co.jp/

●ディジタルサラウンドデコーダLSI

YSS942/941

- AVアンプなどのホームシアター機器で,ドルビーディジタルやDTSなどのディジタルサラウンド方式のすべてを1チップでデコード可能なLSI.
- 自社開発の高精度浮動小数点方式DSPコアを用い,最新の0.15μ微細プロセスを採用.
- 既存のすべてのディジタル音場処理フォーマットのデコードコードをROMでチップ上に搭載.
- 必要な処理を行うワークRAMもすべて内蔵しており,周辺機器を最小限に抑えたコンパクトな実装を可能にしている.
- 動作周波数は180MHzで,ROM/RAMを内蔵することで,高速信号のラインはチップ内部にとどまるため,信号漏れによるノイズが少ない.

■ **ヤマハ(株)**
サンプル価格:¥3,000(YSS942)
¥2,000(YSS941)
TEL:0539-62-5444

●パワーアンプモジュール

ACPM-7813 CDMA/AMPS用パワーアンプ
ACPM-7833 CDMA1900用パワーアンプ
ACPM-7891 EGSM/DCS/PCS用3バンドパワーアンプ

- ACPM-7813/ACPM-7833は,E-pHEMTプロセスにより,40%の電力付加効率を実現することで,小型のバッテリを使用することができ,携帯電話の小型化,薄型化を実現.単一正電源で動作し,待機時の消費電流が低いという特徴をもつ.CDMAに必要な線形動作の特性を,供給電圧が3.2〜4.2Vの間で維持することにより,携帯電話のバッテリ寿命を延長したり,システム設計を簡略化することが可能.
- ACPM-7891は,GSMバンドにおいて60%,DCS/PCSバンドにおいて56%という電力付加効率により,携帯電話の低消費電力化を実現.3〜5.3Vの供給電圧で動作.

■ **アジレント・テクノロジー(株)**
サンプル価格:¥250(100,000個時)
TEL:0120-61-1280

HARD WARE

●ベースバンドダイバーシティ受信機

AD6650

- 独自のスマートパーティショニング方法をベースに設計された，7個の機能をシングルチップ上に搭載したデバイス．
- 既存のソリューションで必要となる5個のアクティブコンポーネントと2個のSAWフィルタが不要となり，プリント基板のサイズおよびテストコストの低減が可能．
- オンボード自動ゲイン制御ループの一機能であるディジタル制御VGA(可変ゲインアンプ)，IFからベースバンドへのI&Q復調器，アナログ低域通過フィルタ機能，2個のA-Dコンバータを集積．
- VCOを内蔵したオンチップ周波数シンセサイザが，I&Q復調器を安定動作させる．
- ディジタルデシメーションフィルタを内蔵しており，これによりシリアル出力データを生成し，同時に当該チャネル外の不要な信号およびノイズを除去．
- IF入力周波数70MHz～300MHzに対応でき，106dB以上という信号直線性領域と110dB以上のダイナミックレンジをもつ．

■ アナログ・デバイセズ(株)
サンプル価格：$20.00(1,000個時)
TEL：03-5402-8129

●グラフィック用メモリ

GDDR2-M

- モバイルPCや高性能PC/WSの画像処理をはじめ，データのエンコードおよびデコードに適する32ビットI/O構成の128Mビットのグラフィック用メモリ．
- 次世代メインメモリとして開発されているDDR2に改良を加え，画像処理用途に最適化．
- Data Inversion Technologyの採用により低ノイズで900Mbps/pin(450MHz)を実現．
- Data Inversion Technologyは，入出力方法を改善することにより，コントローラとメモリの間で発生する信号ノイズを約50%削減し，450MHzの高速動作時でも安定した動作を実現．
- ODT方式をpull-downタイプにすることにより，低消費電力の実現と同時に発熱も抑制．
- ボード上の部品点数を削減できるため，ボード設計にかかる負担を大幅に軽減した．

■ エルピーダメモリ(株)
価格：下記へ問い合わせ
TEL：03-3261-1500
E-mail：info@elpida.com

●TFTカラー液晶表示システム用LSI

HD66777

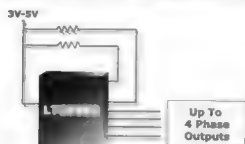
- 1チップで262,000色表示を実現し，メインとサブの液晶画面を駆動可能．
- 132×272ピクセルの画面サイズに対応．
- メイン液晶とサブ液晶でそれぞれ必要だった液晶ドライバを一つにすることで，実装面積を半減，また外付け部品の共有により，部品点数も半減．
- 液晶パネルモジュールの小型化，薄型化を実現し，モジュールコストを同社比で約25%削減できる．
- 動画アプリケーションの補助機能として，背景画面を素通し表示するアルファブレンディング機能や，動画像データへのテキストアイコン表示を実現するオンスクリーンディスプレイ機能を搭載．

■(株)日立製作所
サンプル価格：¥2,200
TEL：03-5201-5226
URL：http://www.hitachisemiconductor.com/jp/

●マルチフェーズスペクトル拡散発振器

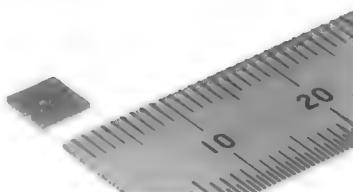
LTC6902

- 1本の外付け抵抗を使用して，5KHz～20MHzの範囲で動作周波数の設定が可能．
- 1.5%の精度と40ppm/°Cの温度安定性を実現．
- スペクトル拡散周波数変調出力が可能で，拡散量が2本目の抵抗で設定されるので，スイッチングレギュレータに起因するEMIの低減に適する．
- 疑似ランダムノイズ技法を使用して，発振器のエネルギを広い周波数帯域に拡散し，電磁放射のピーク値を低下．
- 水晶ベースのソリューションと異なり，50μs～1.5msと起動時間が短く，衝撃や振動に対して高い耐性を備える．

■ リニアテクノロジー(株)
サンプル価格：¥265(1,000個時)
TEL：03-5226-7291 FAX：03-5226-0268
URL：http://www.linear-tech.co.jp/

●光通信システム用外部変調器ドライバIC

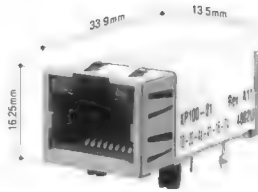
ML0xx18シリーズ

- 高速動作と高耐圧特性を兼ね備えた同社開発のInGaP HBT技術および，高速回路設計技術により，0.1V(両相駆動)の入力必要振幅でも3.0Vの出力振幅が得られる．
- 小型化，低消費電力化，出力駆動振幅が小さいCMOS多重化LSI/物理層LSIを使用した，10Gビット光通信用モジュールの低価格化を可能にする．
- データリタイミング用のフリップフロップを内蔵するため，ジッタの少ない高品位な出力波形が得られる．
- 出力部に50Ω終端抵抗を内蔵することで，光出力波形劣化の要因となるドライバICと光素子間との多重反射を抑制することが可能．

■ 三菱電機(株)
サンプル価格：¥12,000(ML01618)
¥8,400(ML0CP18)
TEL：03-3218-9450

●デバイスサーバ

XPort

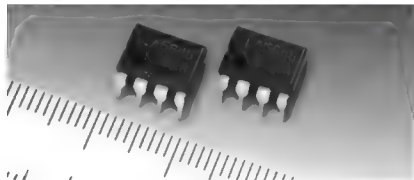
- RJ-45コネクタ内部にCPU，メモリ，Ethernetチップおよびその周辺回路などのハードウェア，OS，TCP/IPプロトコル，Webサーバ，メール発信機能などのソフトウェアを組み込んだ製品．
- インターネット接続が困難であった機器の，ネットワーク接続が可能となる．
- 同製品を搭載した各機器は，個別のホームページを機器内にもつことができるため，遠隔地からのモニタリングなども可能となる．
- 製品のネットワーク化に必要だったプロトコルの移植作業やライセンス契約が不要．
- オプションで，128ビットの暗号機能搭載も可能．

■ 日本ラントロニクス(株)
サンプル価格：¥9,800
TEL：03-3780-7025 FAX：03-3780-7026

HARD WARE

●AC-DC コンバータ用コントローラ IC 開発

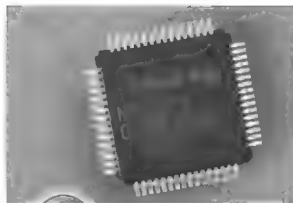
LA5648

- 定格運動時は，ベース巻線電圧を検出することにより，通常の RCC 動作を行う．
- 周波数が外部設定可能な発振器を内蔵しており，軽負荷時に RCC 動作周波数が上昇すると，設定されている内蔵発振器の周波数以上に上昇しなくなり，RCC 動作から他励 PWM 動作に自動移行する．
- 一次側の電流をパルスバイパルスで検出する過電流検出機能と，フォトカプラ経由で二次側電圧の低下を検出してから動作するタイマによって，一定時間経過後発振動作が停止する過電流検出機能をもつ．
- UVLO はロック解除電圧とロック検知電圧のヒステリシスが大きく(約 8V)，外付けの V_{CC}-グラウンド間バックアップコンデンサの容量を小さくできる．

■ 三洋電機(株)
サンプル価格：¥100
TEL：0276-61-8107　FAX：0276-61-8730

●USB2.0 デバイスコントローラ

NET2272

- 米国ネットチップテクノロジー社が開発した，ハイエンド向け 16 ビット汎用 USB2.0 デバイスコントローラ．
- アナログ低電力 USB2.0 トランシーバ，シリアルインターフェースエンジン，三つの 1K バイト双方向バッファ，30MHz PLL，汎用ローカルバスインターフェースを内蔵．
- Dynamic Virtual Endpoint 機能により，物理的なエンドポイントを任意の数の仮想エンドポイントにマッピング可能．
- 広帯域アイソクロンナス転送のサポートや DMA の高速化により，高性能なデータ転送を実現．

■ テクセル(株)
価格：下記へ問い合わせ
TEL：03-5467-9102
E-mail：NetChip@teksel.co.jp
URL：http://www.teksel.com/

●16 ビット デルタ-シグマ型 A-D コンバータ

ADS1605

- 基準電圧内蔵，または外部基準電圧からの入力が可能な，16 ビットデルタ-シグマ型 A-D コンバータ．
- 同社「TMS320C6000」DSP ファミリと接続可能な，簡略化されたパラレルインターフェースを装備．
- フルスケール入力範囲を超える入力信号の検出機能を内蔵．
- 入力信号帯域幅(－3dB)は，2.45MHz．
- 通過帯域リップルは，±0.0025dB 未満(～2.2MHz)．
- アナログ電源は＋5V，ディジタル電源は＋3V．
- ＋2.7V～＋5.25V の範囲で可変のディジタル I/O 電源により，ディジタルインターフェースは多様なロジックファミリに対応．
- 消費電力は外付け抵抗で設定でき，低速動作時には 315mW～570mW の範囲で可変．
- 未使用時には I/O ピンによるパワーダウンモードの設定が可能．

■ 日本テキサス・インスツルメンツ(株)
価格：¥4,167(1,000 個時)
FAX：0120-81-0036
URL：http://www.tij.co.jp/pic/

●DSP

TMS320C6416/ TMS320C6415/ TMS320C6414

- 「TMS320C64x」DSP コアをベースに開発され，130nm 銅配線プロセステクノロジにより，動作周波数 720MHz を実現．
- 1M バイトの高速メモリ，3G ワイヤレスアクセラレーションコプロセッサならびに高速な周辺回路を内蔵し，アプリケーションおよびリアルタイムのデータ処理を高速化．
- 「TMS320C6414」は，システムメモリからの数 G バイト/s の高速データ転送を管理し，64 チャネルの EDMA コントローラを搭載．128 個の TDM チャネルをサポートする McBSP，AC97 および IIS オーディオインターフェースを 3 組内蔵．
- 「TMS320C6415」は，プロセッサ間の通信制御のため，33MHz，32 ビットの PCI および HPI に接続機能．50MHz の Utopia Level Ⅱ ATM 接続機能を内蔵．

■ 日本テキサス・インスツルメンツ(株)
価格：$199.89(C6414/10,000 個時)
$264.49(C6415/10,000 個時)
$279.89(C6416/10,000 個時)
FAX：0120-81-0036
URL：http://www.tij.co.jp/pic/

●電圧レギュレータ

TPS795xx/ TPS796xx/ TPS786xx

- RF レシーバおよびトランスミッタ，VCO，オーディオアンプなど，ノイズに敏感な回路の電源向けに低ノイズ，高電源リップル除去比，高速の過渡応答特性を提供．
- 出力電流 500mA，1A，1.5A の製品が用意され，5 または 3.3V 分散型 BUS 動作のアプリケーションをサポート．
- 10KHz において最大 53dB の PSRR，100Hz～100KHz において 33μVrms の低ノイズならびに過渡応答特性をもち，ノイズに敏感なアプリケーションに適する．
- スタンバイモード時の消費電流を 1μA 未満まで低減でき，ポータブルアプリケーションのバッテリ寿命の延長が可能．
- 出力電圧固定および，1.2V～5.5V の範囲で可変の製品を用意．
- 同社の BiCMOS プロセスで製造され，出力電流 1A の場合に，250mV の低いドロップアウト電圧を実現．

■ 日本テキサス・インスツルメンツ(株)
価格：$0.96(TPS795xx/1,000 個時)
$1.04(TPS796xx/1,000 個時)
$1.28(TPS786xx/1,000 個時)
FAX：0120-81-0036
URL：http://www.tij.co.jp/pic/

●インサーキットエミュレータ

MC9S12T64 対応エミュレータ

- 米国 Nohau 社が開発した，モトローラ製 MC9S12T64 対応のエミュレータ．
- CPU インサーキット方式のフル ICE と BDM 接続による BDM エミュレータの 2 種類を用意．
- フル ICE は，1M バイトエミュレーションメモリやフルバストレース，マイコン内蔵フラッシュメモリへのプログラミング機能，メモリ内容をリアルタイムで参照するシャドウ RAM 機能などを装備．MC9S12T64 がもつ COP ウォッチドッグ，STOP/WAIT モード，self-clock モードなどの特殊動作モードやリセットモードなどをサポート．
- BDM エミュレータは，マイコン内蔵フラッシュメモリへのプログラミング機能やシャドウ RAM 機能などを装備した，簡易エミュレータ．
- エミュレータ/ハードウェアの機能を最大限に引き出す，C/C++ 対応高級言語デバッガ「Seehau」を標準添付．

■(株)ソフィアシステムズ
価格：¥1,600,000～(フル ICE)
¥458,000(BDM エミュレータ)
TEL：044-989-7245　FAX：044-989-7005
E-mail：market@sophia-systems.co.jp
URL：http://www.sophia-systems.co.jp/

●インサーキットエミュレータ

UniSTAC II for PXA255

- インテル PXA25x アプリケーションプロセッサを採用した，組み込みシステムのハード設計およびソフト開発をサポートするフルICE．
- エミュレーションメモリ（最大16Mバイト）やフラッシュROMへプログラムを直接ダウンロードする機能，64Kフレームのリアルタイムフルバストレース（SDRAMサポート），カバレッジ/パフォーマンス/プロファイル機能などを標準で提供．
- 五つのトリガポイントは，トリガオプションやデータフィルタリング機能などと組み合わせて使用することが可能．
- テストターゲットを利用することで，ハードウェアの完成前にプログラムのダウンロードや実行が可能．
- ICE専用のコネクタをターゲットボードのレイアウトにデザインすることで，ターゲットプロセッサを取り外さなくても，ICEを接続することが可能．

■（株）ソフィアシステムズ
価格：¥798,000（USB対応JTAG ICE）
¥980,000（USB＋LAN対応JTAG ICE）
¥1,980,000（USB＋LAN対応フルICE）
TEL：044-989-7245　FAX：044-989-7005
E-mail：market@sophia-systems.co.jp
URL：http://www.sophia-systems.co.jp/

●DSP評価ボード

G6205AVIO plus

- テキサス・インスツルメンツ社製DSP「TMS320C6205」を搭載した評価ボードに，映像，音声の入出力を加え，この評価ボード1枚だけで映像，音声のDSPアルゴリズム，コントロールアプリケーションの開発評価が可能．
- ボードに実装されているAV入出力デバイスへのインターフェース部分については，DSPのソースコードを添付．オリジナルCODECなどの評価を短時間で実行可能．
- EVAボードの回路図が添付されており，オリジナル製品設計において，DSP周辺部の設計資料として利用可能．
- PCIバスボード型評価ボードなので，PCでの開発が容易．
- Motion-JPEG，G.711サンプルソフトが付属し，その他H.263，MPEG-4などの各種ソフトIPを用意．

■ガイオ・テクノロジー（株）
価格：下記へ問い合わせ
TEL：045-450-5691　FAX：045-451-5697
E-mail：dsp_kit@gaio.co.jp
URL：http://www.gaio.co.jp/

●FPGA用スタータキット

ProASIC Plus スターターキット

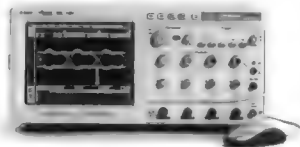
- ProASIC Plusデバイス「APA075」および評価ボード，Liberoゴールド統合設計環境，低価格プログラマ，プログラミングケーブル，電源，チュートリアル，サポートマニュアルで構成．
- 不揮発性，低消費電力FPGAのISP（インシステムプログラマビリティ），I/O，FlashLockオンチップセキュリティシステムなどを試用することが可能．
- Liberoゴールド統合設計環境では，モデルテクノロジー社のModelSimシミュレーションおよび設計検証ソフトウェア，シンプリシティ社のSynplify合成ソフトウェア，アクテル社のDesignerシリーズ配置配線ソフトウェアを利用することができる．

■アクテルジャパン（株）
価格：$249
TEL：03-3445-7671　FAX：03-3445-7668
URL：http://www.actel.com/

●広帯域リアルタイムオシロスコープ

Infiniium 54853A

- 2.5GHzのアナログ帯域に対応高速ディジタルの汎用回路設計に最適．
- 20GS/sのA-Dコンバータを4個搭載しているため，4チャネル同時に使用しても最大サンプル速度を維持できる．
- 一つのプローブでシングルエンドプローブにも差動プローブにも対応する「Infinii Maxプローブシステム」に対応．
- 独自開発のメモリ長自動調整機能をもったメモリ管理用ASIC「MegaZoom」を搭載．サンプリング速度5Gsps以上のときは各チャネル1Mバイト，2Gsps以下のときは32Mバイトのメモリを使用することができる．ロングメモリを採用した場合でも，オシロスコープの画面更新速度や反応速度が遅くなることはない．

■アジレント・テクノロジー（株）
予定価格：¥4,400,000～
TEL：0120-421-345

●広帯域差動プローブ

P7350型

- P7330型3.5GHz差動プローブの上位機種として，5GHz帯域を備え，同社のオシロスコープで使用することにより高いCMRRを確保し，より広帯域での差動測定を実現．
- ダイナミックレンジを±2.5Vに向上し，プローブ寸法，形状はP7330型と同一．
- 100K Ωの高差動入力抵抗，0.3pF以下の低差動入力容量により，プローブによる影響を最小にできるため，再現性のよい差動測定が可能となる．
- PCI Express，InfiniBand，シリアルATA，シリアルRapidIOなどの高速差動信号測定に適する．

■日本テクトロニクス（株）
価格：¥998,000
TEL：03-3448-3010　FAX：0120-046-011
URL：http://www.tektronix.co.jp/

●モバイルVPN商品

CX7500シリーズ

- ノートPCなどモバイル端末の固定IPアドレスのまま，社外の無線LANなどのアクセスポイントからインターネット経由で自社網にアクセスが可能．
- 無線LANアクセスポイントから離れても，移動を意識せずにPHSや携帯電話，有線などでのアクセス方法に自動的に切り替えることが可能な，シームレスローミングサービスを提供．
- VPNとしてIPSec機能をサポートすることにより，ローミングクライアントとホームエージェント間をトンネリングしてユーザーデータを保護するため，異なるネットワーク間でもセキュリティを確保した通信が可能．

■日本電気（株）
価格：¥2,350,000～（CX7504-HA）
¥1,560,000～（CX7504-HS）
¥34,000～（CX7504-RC，10クライアント）
TEL：03-3798-1428
E-mail：info@bpd.jp.nec.com

SOFT WARE

●多漢字GUIシステム

PMC T-Shell

- 標準T-Engine仕様に準拠したT-Engine/SH7727，T-Engine/VR5500，T-Engine/ARM720-S1C上で動作するGUIミドルウェア．
- グラフィック画面に図形や文字を描画するディスプレイプリミティブ，画面上のパーツ類(テキストボックスやスイッチなど)，メニュー，ウィンドウなどを管理するGUIマネージャ，VJEを用いたかな漢字変換機能，多漢字の表示機能および17万字の多漢字，多言語用フォント，TCP/IPなどのほか，タッチパネルなどのHMI画面の設計に適したビジュアル言語「マイクロスクリプト」が含まれる．
- 動作するCPUに応じて「T-Engine/SH7727開発キット」，「T-Engine/VR5500開発キット」，「T-Engine/ARM720-S1C開発キット」の3種類を用意．
- マイクロスクリプトを利用すれば，再コンパイルも不要となり，CPUに依存しないプログラムの作成が可能．

■ パーソナルメディア(株)
価格：下記へ問い合わせ
TEL：03-5702-7858
E-mail：sales@personal-media.co.jp
URL：http://www.personal-media.co.jp/te/

●汎用有限要素法解析ツール

ANSYSプロダクト7.0

- 幅広い物理現象の解析に対応するマルチフィジックス機能を拡充してきた汎用有限要素法解析ツールANSYSシリーズと，各種3次元CADとの連携を推進してきた解析ツールDesignSpaceを一つの製品ラインに統合．
- 結果表示に利用可能な座標系を追加．直交座標系，円筒座標系が利用可能な座標系作成機能を搭載．
- パーツ，面，辺，頂点群に対するグループ設定が可能．
- ツリーのリスト表示機能をもつワークシートタブ．
- 応答パラメータとして反力の取り扱いが可能．
- 解析結果項目をExcelファイルとして出力可能．
- データベースからメッシュと結果情報を削除し，ファイルサイズを低減化した．

■ サイバネットシステム(株)
価格：¥550,000～
TEL：03-5978-5420　FAX：03-5978-5960
E-mail：anasales@cybernet.co.jp

●RTLプロトタイピングソフトウェア

Certifyソフトウェア Ver.6.2

- ゲーテッドクロックのレポート機能やソースレベルでのパーティショニングなどにより，プロトタイピングプロセスの可視性を向上するASIC検証機能を提供．
- 新型タイミングエンジンおよびタイミング解析機能を追加搭載し，新たにアルテラ社の高性能デバイスStratixをサポート．
- 対応OSはUNIX(SolarisおよびHP)，Windows NT/2000/XP Proに加えLinuxを追加．
- 論理階層をサポートするMultiPointシンセシステクノロジで容量を拡大．
- シノプス社のDesignWareコンポーネントをVerilog-HDL/VHDLの双方でサポート．
- Verilog-2001をサポート．
- ザイリンクス社のChipScope Proをサポート．
- 「Amplify Physical Optimaizer」ソフトウェアとの統合を強化し，タイミングを最適化するために，パーティショニング後のCertifyソフトウェアからの出力結果をダイレクトにAmplifyソフトウェアに渡すことが可能．

■ シンプリシティ(株)
価格：¥15,000,000
TEL：03-5358-3311

●SystemCデバッガソフトウェア

XModelink SystemC Debugger

- SystemCデバッグとマイクロソフト社のC++開発環境「VisualC++.NET」の双方への対応機能をもつ．
- FPGAなどへ実装するハード/ソフト混在型のシステム全体設計，検証機能を搭載．
- シミュレーションの時間単位でのブレーク機能．
- sc_lv型，sc_out型などSystemC特有の変数やポート値の直接表示や値代入機能をサポート．
- 並列に動作するSystemCのプロセスを同時に表示させながらのデバッグ機能を搭載．
- トランザクションレベルのイベント同期波形ビューア機能を搭載．
- VCD形式などでの信号波形出力時の全ノード自動トレース機能．
- Windows XP/2000上で動作．

■ キャッツ(株)
価格：¥980,000
TEL：045-473-2816　FAX：045-473-2673
E-mail：info@zipc.com

●RSA SecurIDの認証サーバ

RSA ACE/Server 5.1

- ユーザー情報を格納するLDAPディレクトリサーバとの同期機能を強化し，LDAPディレクトリをユーザー情報資源として利用可能．
- 認証処理の中心として稼動しているプライマリサーバで障害が発生した際，そのサーバ名やIPアドレスをレプリカサーバへ変更するためのプロセスを提供．
- 認証処理機能回復の時間を短縮．
- SecurIDの新規，交換の要求，SecurIDの割り当てやアクティブ化などの手続きや作業をワークフロー化したWebアプリケーションである「RSA SecurID Web Express」をサポート．
- SecurID導入のプロセス全体が自動化され，配布工数を従来の7ステップから3ステップに削減．
- SecurID認証エンジンに，米国政府の標準暗号化方式であるAESを採用した「RSA SecurID AESトークン」をサポート．

■ RSAセキュリティ(株)
価格：¥719,000～(25ユーザー～)
TEL：03-5222-5230　FAX：03-5222-5271
E-mail：info-j@rsasecurity.com
URL：http://www.rsasecurity.co.jp/

●マルチメディア拡張モジュール

NetFront v3.0 Multimedia Extension

- NetFront v3.0にSMIL機能を拡張実装し，テキストやイメージ，音声，動画，アニメーションなどを統合して扱うことが可能．
- SMILの同期機能で，ムービーのようなマルチメディアコンテンツの自動再生も可能．
- SVGビューアを拡張実装することで，SVGベクタ画像コンテンツをブラウザ上で取り扱うことが可能．コンテンツ自体を多種多様な端末の画面サイズに最適化して表示することや，単一画面上でのSVGベクタ画像コンテンツの拡大縮小表示が可能．
- NetFrontブラウザ本体とリソースを共有できるため，携帯電話のようなハードウェアリソースの限られた情報家電に組み込むことが可能．
- SMILプレーヤ，SVGビューアはそれぞれ単独での拡張搭載が可能で，ニーズにあった組み合わせで搭載することが可能．
- SMILプレーヤはSMIL Basicに，SVGビューアはSVG Tinyにそれぞれ準拠．

■ (株)ACCESS
価格：下記へ問い合わせ
TEL：03-5259-3685　FAX：03-3233-0222
E-mail：prinfo@access.co.jp

SOFT WARE

●2D/3D CAD ソフト

TURBOCAD v8 シリーズ

- Microsoft Office インターフェースに準拠した，400以上の作図，編集，表示ツールをサポート．
- 壁ツール，ポイントハッチング，フォーマットペインタ，雲形など，作画の効率をアップする機能を搭載．
- デザインディレクター，パートツリーパレットを搭載．
- ACIS v6 ソリッドモデリングの機能強化．
- LightWorks レンダリングによる，ラジオシティレイトレーシングの強化．
- マテリアルパレット，ルミナンスパレット，環境マップパレットを新たに搭載．

■ キヤノンシステムソリューションズ(株)
価格：¥78,000(Professional)
¥38,000(Standard)
¥12,800(TURBOSketch v8)
TEL：03-5815-7258
E-mail：tcad-info@canon-sol.co.jp

●開発ツール

eBinder for V850

- NEC エレクトロニクス社製の 32 ビット RISC マイコン V850 シリーズ対応の「eBinder」．
- リアルタイムデバッグツール，システム解析ツールを含めたマルチプログラミングツール群およびソフトウェア部品構築支援ツールなどを提供．
- μITRON アプリケーション開発の期間とコストを削減．
- RX850Pro ベースのアプリケーション開発をサポート．
- デバッグポートとして高速 Ethernet をサポートするため，ストレスなくデバッグが可能．
- V850E/MA1 搭載 SolutionGear ボードに対応した BSP，イーソル製の FAT ファイルシステム「PrFILE」，TCP/IP プロトコルスタック「PrCONNECT」を搭載．
- NEC エレクトロニクス社の純正 C コンパイラ「CA850」に対応．

■ イーソル(株)
価格：下記へ問い合わせ
TEL：03-5301-5325　FAX：03-5376-2538
E-mail：cp-inq@esol.co.jp
URL：http://www.esol.co.jp/embedded/

●TCP/IP プロトコルスタック

KASAGO for T-Engine

- エルミックシステム社が開発した，T-Engine 上で動作するインターネット接続用ソフトウェアの TCP/IP(IPv4/IPv6)プロトコルスタック．
- T-Engine に搭載される標準リアルタイム OS「T-Kernel」および T-Kernel 上で動作するミドルウェアの標準流通形式「T-Format」に対応．
- 組み込みシステム向けに設計，開発されており，高速，コンパクト，高信頼性を実現．
- 従来のソフトウェアとの高い互換性を提供する IPv6/IPv4 デュアルスタック構造を採用．
- IPSec や H.323 をはじめとする豊富なオプション製品群をサポート．

■ パーソナルメディア(株)
価格：¥150,000
TEL：03-5702-7858
E-mail：sales@personal-media.co.jp
URL：http://www.personal-media.co.jp/

●暗号セキュリティソリューション

CipherCraft シリーズ

- 国産の暗号アルゴリズム「Camellia」に対応した暗号通信 VPN ソリューション．
- Camellia のほかに，3-DES などの複数の暗号にも対応．
- インターネットを利用して専用線並みのプライベートネットワークを構築するため，ランニングコストの大幅な削減が可能．
- 既存システムに依存することなく導入ができるため，導入コストを安価にすることが可能．
- 通信元と通信先のアプリケーションの間に独自の通信経路を設定し，TCP レベルで暗号通信を行うため，既存のハードウェアやソフトウェアを新規購入または更改することなく，既存システムにセキュアな暗号通信システムを導入可能．
- 暗号通信を利用できるマシンをあらかじめ制限することによって，簡易ファイアウォールとして利用することも可能．

■ NTT ソフトウェア(株)
価格：¥5,000,000
TEL：045-212-7421
E-mail：ccraft-vpn@cs.ntts.co.jp

●翻訳ソフトウェア

クロスランゲージ 多言語フルパック 2 for Windows

- 日英・英日，中日・日中，韓日・日韓およびヨーロッパ言語翻訳の四つのソフトウェアを同梱し，日本語と 11 か国語を双方向で翻訳可能．
- 多言語 OCR ソフトウェア「Asian Reader」は，日本語，中国語，韓国語，ロシア語，英語の文字認識が可能．パターン認識に加え，各国語のスペルチェック機能を搭載．
- 研究社「リーダーズ＋プラス V2」および吉林堂「簡体字 日中/中日辞典」の辞書引きが可能．
- 日本語，中国語，英語の各 Windows に対応した多言語ワープロ「Asian Writer」を搭載．日本語，中国語(簡単字，繁体字)，韓国語，欧州 14 か国語(ロシア，ヘブライ語を含む)の入力が可能．Unicode に対応しているため，多言語が混在した文書の編集が可能．
- 対応 OS は WindowsXP/Me/2000/98SE．

■ (株)クロスランゲージ
価格：¥198,000
TEL：03-5287-7588
E-mail：info@crosslanguage.co.jp
URL：http://www.crosslanguage.co.jp/

●翻訳ソフトウェア

PC-Transer V10

- 研究社『リーダーズ＋プラス V2』，『リーダーズ英和辞典(第 2 版)』および『リーダーズ・プラス』を搭載．
- 使用頻度の高い表現を文章ごとに登録し，人による翻訳資産と機械翻訳を融合させる技術である翻訳メモリ機能を搭載．登録文と主語のみが変わるなど，文の一部が変更されている場合でも，正しい機械翻訳を行うことができる．
- 契約書関連の表現例 300 例を含む，ビジネスで使う表現例を 45,000 例を搭載した基本翻訳メモリを標準搭載．
- 日英対訳エディタの英文側に，ユーザー辞書内の訳語の挿入が可能．独自の用語をあらかじめユーザー辞書に登録しておき，機械翻訳を行う前にユーザー辞書の訳語を原文に挿入しておくことで，用語の統一などをはかることができる．

■ (株)アスキーソリューションズ
価格：¥68,000～(スタンダード版)
¥98,000～(プロフェッショナル版)
¥298,000～(エンタープライズ版)
TEL：03-4524-6001
E-mail：retail@asciisolutions.com
URL：http://www.asciisolutions.com/

海外・国内イベント/セミナー情報

INFORMATION

海外イベント

4/27-5/2 **NETWORLD+INTEROP Las Vegas 2003**
Las Vegas Convention Center, Las Vegas, LA, USA
Key3Media
http://www.interop.com/lasvegas2003/

5/11-15 **International Conference for Communications**
Egan Convention Center, Anchorage, AL, USA
IEEE
http://www.icc2003.com/

5/13-16 **Electronic Entertainment Exposition**
Los Angels Convention Center, Los Angels, CA, USA
Electronic Entertainment Expo
http://www.e3expo.com/

5/25-28 **International Symposium on Circuits and Systems**
Imperial Queen's Park Hotel, Bangkok, Thailand
IEEE
http://www.iscas2003.org/

6/2-6 **COMPUTEX TAIPEI**
Taipei World Trade Center, Taipei, Taiwan
CETRA
http://www.taipeitradeshows.com.tw/computex/

6/3-5 **Infosecurity Canada**
Sheraton Centre Hotel, Toronto, Ontario, Canada
Reed Exhibitions
http://reedexpo.ca/infosec/

6/17-19 **International Conference on Consumer Electronics**
Lax Marriott Hotel, Los Angeles, CA, USA
IEEE
http://www.icce.org/

※先月号に掲載したSEMICON Singapore 2003は，SARSの影響により8/12～8/14に日程が変更されました．

国内イベント

5/14-16 **2003 実装プロセステクノロジー展**
日本コンベンションセンター(幕張メッセ，千葉県千葉市)
(社)日本ロボット工業会
http://protec.jesa.or.jp/jp/frontpage/index.html

5/20-23 **ビジネスシヨウ 2003**
東京国際展示場(東京ビッグサイト，東京都江東区)
(社)日本経営協会
http://bs.noma.or.jp/

5/21-23 **LinuxWorld Expo/Tokyo 2003**
東京国際展示場(東京ビッグサイト，東京都江東区)
IDGジャパン
http://www.idg.co.jp/expo/lw/

5/27-30 **INFORMATION STORAGE WEEK 2003**
東京ファッションタウン(TFT)ビル(東京都江東区)
国際ディスクドライブ協会 IDEMA Japan
http://www.idema.gr.jp/isw2003.htm

6/3-4 **RSA Conference 2003**
東京国際フォーラム(東京都千代田区)
Key3Media
http://www.key3media.co.jp/rsa2003/

6/4-6 **JPCA Show 2003 (国際電子回路工業展)**
東京国際展示場(東京ビッグサイト，東京都江東区)
(社)日本プリント回路工業会
http://www.jpca.jp/2003/index.html

6/4-6 **ビジネスシヨウ OSAKA 2003**
インテックス大阪(大阪府)
(社)日本経営協会
http://www.noma.or.jp/bsosaka/

開催日，イベント名，開催地，問い合わせ先の順

セミナー情報

プロジェクト管理者のためのアジャイル開発プロセス技術解説とプロジェクト管理法
開催日時　：5月8日(木)～5月9日(金)
開催場所　：SRCセミナールーム(東京都高田馬場)
受講料　：76,000円
問い合わせ先：(株)ソフト・リサーチ・センター，☎(03)5272-6071
http://www.src-j.com/seminar_no/23/23_057.htm

C言語ポインタ徹底習得(ポインタを正しく教える方法)
開催日時　：5月9日(金)
開催場所　：CQ出版セミナールーム
受講料　：13,000円
問い合わせ先：エレクトロニクス・セミナー事務局，☎(03)5395-2125

ビギナのためのアナログ回路設計
開催日時　：5月10日(土)
開催場所　：CQ出版セミナールーム
受講料　：3,000円
問い合わせ先：エレクトロニクス・セミナー事務局，☎(03)5395-2125

TCP/IPプロトコル技術解説講座[基礎編]
開催日時　：5月12日(月)～5月13日(火)
開催場所　：SRCセミナールーム(東京都高田馬場)
受講料　：78,800円(サブテキスト2,800円含む)
問い合わせ先：(株)ソフト・リサーチ・センター，☎(03)5272-6071
http://www.src-j.com/teiki_no/Src/tcp_1.htm

USB2.0仕様解説
開催日時　：5月14日(水)
開催場所　：オームビル(東京都千代田区)
受講料　：58,500円(1口で1社3名まで受講可)
問い合わせ先：(株)トリケップス，☎(03)3294-2547，FAX(03)3293-5831
http://www.catnet.ne.jp/triceps/sem/c030514a1.htm

制御用コンピュータI/O操作プログラミング
開催日時　：5月14日(水)～5月16日(金)
開催場所　：高度ポリテクセンター(千葉県千葉市)
受講料　：30,000円
問い合わせ先：雇用・能力開発機構 高度ポリテクセンター事業課，☎(043)296-2582
http://www.apc.ehdo.go.jp/

はじめてのHDL
開催日時　：5月16日(金)
開催場所　：CQ出版セミナールーム
受講料　：13,000円
問い合わせ先：エレクトロニクス・セミナー事務局，☎(03)5395-2125

ステートマシンの設計技術
開催日時　：5月17日(土)
開催場所　：CQ出版セミナールーム
受講料　：13,000円
問い合わせ先：エレクトロニクス・セミナー事務局，☎(03)5395-2125

VoIPの最新動向とネットワーク構築手法
開催日時　：5月19日(月)
開催場所　：オームビル(東京都千代田区)
受講料　：58,500円(1口で1社3名まで受講可)
問い合わせ先：(株)トリケップス，☎(03)3294-2547，FAX(03)3293-5831
http://www.catnet.ne.jp/triceps/sem/c030519a.htm

シングル・サインオン認証標準技術SAMLの基礎技術解説と応用技術解説
開催日時　：5月19日(月)～5月20日(火)
開催場所　：SRCセミナールーム(東京都高田馬場)
受講料　：76,000円
問い合わせ先：(株)ソフト・リサーチ・センター，☎(03)5272-6071
http://www.src-j.com/seminar_no/23/23_091.htm

CMOSアナログ回路入門
開催日時　：5月22日(木)
開催場所　：CQ出版セミナールーム
受講料　：13,000円
問い合わせ先：エレクトロニクス・セミナー事務局，☎(03)5395-2125

WindowsCEプリントシステムセミナー
開催日時　：5月27日(火)
開催場所　：みなとみらい2-3-3クイーンズタワーB 7Fクイーンズフォーラム会議室(横浜市西区)
受講料　：無料
問い合わせ先：(株)グレープシステム基本ソフトウェア事業部セミナー係，☎(045)222-3761，FAX(045)222-3759
http://www.grape.co.jp/seminar.html

パソコン・リアルタイムOSプログラミング技法
開催日時　：5月27日(火)～5月29日(木)
開催場所　：高度ポリテクセンター(千葉県千葉市)
受講料　：35,000円
問い合わせ先：雇用・能力開発機構 高度ポリテクセンター事業課，☎(043)296-2582
http://www.apc.ehdo.go.jp/

わかりやすい情報通信ネットワーク
開催日時　：5月29日(木)
開催場所　：NTTアドバンステクノロジ会議室(東京都)
受講料　：19,000円
問い合わせ先：サイペック(株)，info@r-sipec.jp
http://www.rlz.co.jp/seminar/seminardata.php?id=RGS305802

日程はすべて予定です．問い合わせ先にご確認のうえ，お出かけください．

読者の広場

2003年4月号特集「解説！USB徹底活用技法」に関して

▷USB2.0の特集は参考になりました．とくにコラムの「なぜ現在のUSB2.0は480Mbpsの実力が出ないのか？」は，インテルのICH4の性能が良いといわれているのを不思議に思っていたので，詳しい解析があり興味深かったです．また，Windowsドライバだけでなく，Linuxなどのドライバのチューニングはどうなのだろうかと興味がわきました．　　　（玉出のタマ）

▷組み込みマイコンでホスト機能をもったUSBコントローラを心待ちにしていたのですが，ついに出現したかという気持ちです．これにより，組み込みシステム内においてUSBデータ転送が可能になるので，たいへんうれしく思っています．その反面，いわゆるレガシーI/O（シリアルなど）がすたれていくのは残念です．　　　（白石 隆）

▷USB2.0のPCが常識になってきた現在，今月号の特集はタイムリーで良かったです．USBでどこまで何を実装するかおもしろいテーマで，そのようなことを考えるとき，ハンドブック的に使える中身の濃い特集でした．　　　（JR9JUK）

▷「解説！USB徹底活用技法」はたいへん参考になりました．とくにUSB2.0対応や高速転送対応に関する記事を興味深く読むことができました．　　　（福沢陽介）

▷USBはいいですね．しかしWindowsはUSBがあたりまえなのに，未だにIDEからの起動が主だからね．BIOSもUSBストレージを自動認識してブータブル対応にしてほしいね．USBのCDドライブをもっているのに，インストールのためにCDドライブを換えるのはたまらないからね．　　　（てんろん）

その他

▷オフコンメーカーからの二千年問題修正見積もりが一千万円以上というのを，安いと見る人もいるでしょう．旧データの利用を考えたら大丈夫かなと疑問です．でも，その中小企業の社長の判断は正しい．

ところで，2004年も危険年（初めての閏年）だと知っている人は少ないでしょう．プログラムの修正はかなり危ないんです．　　　（アロゴン）

▷JPEG2000のからくりがなかなかピンときませんでしたが，「JPEG2000デコーダをDSPへ実装する」の前半の解説で，少しピンときました．なにせ頭がかたいもので…….技術者として限界が？ 48歳の春．　　　（star）

▷USBインターフェースを用いたシステムの開発をしようとしていたのでタイムリーな特集でした．

来月号の組み込みシステム開発技法にも期待しています．長年組み込み系のソフトを開発してきましたが，開発技法にまで気を配っているところが少なく，他のソフトウェア開発プロジェクトに比べてひけを感じることが多かったからです．　　　（ユースケース サンタマリア）

特集担当デスクから

☆今回の特集は，TCP/IPとVoIPの2本建てである．第1章のTCP/IPは，日常生活にすっかり溶け込んだ感のあるインターネットの基礎技術である．実際にTCP/IPを使ったプログラミングを行っている読者も多いと思われるが，ここは基本に立ち戻って知識を再確認していただきたい．

☆第2章以降は，TCP/IPの応用例として近年脚光を浴びているVoIPである．音声を送受信する“だけ”であれば，極論すれば.wavファイルをftpで送受信してもかまわない．しかし，それを電話のレベルにまで引き上げることは容易ではない．もともとTCP/IPはリアルタイム性を第一に考えた規格ではなかった．しかし，さまざまな手段を用いることにより，一般的な音声通話が可能なレベルにまで仕上げている．枯れた技術の上に新しい技術を載せて有効活用している良い例ではないだろうか．

☆また，VoIPにおけるビジネスモデルに関しても執筆していただいた．インターネット＝すべて無料というユーザーの考えを覆すのは難しいかもしれない．VoIPをビジネスに結び付けるのも，今後の課題である．

☆ところで，筆者の方々と話していて気付いたのだが，VoIPは「ヴォイプ」と読むのが一般的なようだ．ヴォイス+IPであるから，そう読むのが正しいのであろう．

Interface

次号予告 高速バスシステム徹底研究

2003年7月号は5月24日発売です

LV-TTL/HSTL/LVDS/HyperTransport/PCI Express/PCI-X/GigabitEthernet/IEEE1394.b/USB2.0

最近ではLVDSなど差動伝送に対応したFPGAなどが登場し，数百Mbpsから Gbps程度のデータ通信を実現できるようになってきた．さらに，USBやEthernetのようなインターフェースからPC/AT互換機のメモリバスやCPUのFSBなど，あらゆるバス/インターフェースは高速化の一途をたどっている．これらはデュアルエッジでデータを転送したり，低電圧化/差動駆動化といった電気的な改良などの積み重ねにより実現されている．

またバスの転送帯域を上げるためにバス幅を広げる方法では，配線遅延などの問題でクロックに対してデータの到着がばらついてしまうため高クロック化が難しい．高速化を実現するためにシリアル化，または狭バス幅化を採用するバスシステムも増えてきている．

そこで次号では，「高速バスシステム」というキーワードで，デバイス間通信，基板間通信，筐体間通信，システム間通信など，用途や距離に応じたそれぞれの分野でよく使われている，またはこれから普及すると思われるバスインターフェースを取り上げ，どのようにして高速化を実現しているかなどを解説する．

編集後記

■この欄を書いている現在，「イラク戦争」が続いています．戦争で技術開発が進み，軍需産業が潤い……という話もあるけれど，個人としては釈然としません．現実は複雑な事情の積み重ねであるにしても，戦争が生むのは悲しみばかり．人は，もっと建設的なテーマに向けて，自分のベストを尽くすべきではないでしょうか？　(洋)

■いまさら述べるのもおかしな話だが，「急所」といわれている場所は，本当に危険だから「急所」と称されているのだと実感．何が起こったのかというと，いわゆる男がもっとも痛みを感じるとされている部分に強烈な打撃をもらってしまったのだ．これを書いている時点で二日が経っているが，未だに気持ちが悪い．気を付けましょう．　(＝IO)

■運転免許証の更新期間っていつの間にか誕生日を挟んで2か月間になってたんですね．更新のお知らせのハガキが来て知りました．前々回までは誕生日を過ぎて(失効して)から更新してたので，無事故無違反でもゴールドになりませんでしたが，前回はちゃんと誕生日前に更新したので，次ちゃんと更新すればついにゴールドです！(M)

■HDD+DVD複合ビデオデッキを買ってから早3か月，まだ1枚もDVDを焼いていません．もうこれ以上，物を増やしたくないので，見ては消しの毎日です．この調子で物を捨てられる性格に改善したいのですが……せめてネットワーク上へ置ければなぁ(結局捨てられない)．物欲を絶ちたい今日このごろ．もう限界．　(み)

■イラク戦争が始まって一週間，いろいろ言いたいことは頭に浮かんでくるが，何を言っても意味を見出せないような複雑な気分である．正義や人命尊重という言葉がむなしい．日本は平和ボケなのかもしれないが，戦争という状況から遠いところにいられるのは幸せだと思う．その代償が米国全面支持になるのだが，安全はいつまでも続くのだろうか．　(Y)

■気がつくと，高田馬場駅の発車音がいつの間にか「鉄腕アトム」の音楽でした．後で知ったのですが，アトムは2003年4月7日に高田馬場で誕生した設定だそうです．日々技術は進歩しています．それでも手塚治虫が思い描いた未来には追いつけなかったのですが，いつかきっとアトムのようなロボットが見られる時がくるんでしょうね．(Y2)

■キダム日本公演，もうご覧になりましたか？　超技の連発に驚き，涙し，最後には感動の嵐．開幕以降，友人と夜通し熱く語り，見終わった後は終電まで飲み会になるので疲労もたまる．私は1週間見ないと手が震えてくる．こうなると中毒．治療法は生の舞台を見て感動するしかないらしい．　(太陽熱)

■キムタク主演のドラマ「GOOD LUCK」も高視聴率を記録して終わりました．予想されたことではありますが，パイロットの人気が急上昇．航空会社では採用応募が急増すると同時に株価も上昇．不況に強いキムタク効果．依然衰える気配はありません．今度のドラマは株価にどれだけ貢献できるかということになるのでしょうか．(5)

お知らせ

▶読者の広場

本誌に関するご意見・ご希望などを，綴じ込みのハガキでお寄せください．読者の広場への掲載分には粗品を進呈いたします．なお，掲載に際しては表現の一部を変更させていただくことがありますので，あらかじめご了承ください．

▶投稿歓迎

本誌に投稿をご希望の方は，連絡先(自宅/勤務先)を明記のうえ，テーマ，内容の概要をレポート用紙1～2枚にまとめて「Interface投稿係」までご送付ください．メールでお送りいただいても結構です(送り先はsupportinter@cqpub.co.jpまで)．追って採否をお知らせいたします．なお，採用分には小社規定の原稿料をお支払いいたします．

▶本誌掲載記事についてのご注意

本誌掲載記事には著作権があり，示されている技術には工業所有権が確立されている場合があります．したがって，個人で利用される場合以外は，所有者の許諾が必要です．また，掲載された回路，技術，プログラムなどを利用して生じたトラブルについては，小社ならびに著作権者は責任を負いかねますので，ご了承ください．

本誌掲載記事をCQ出版(株)の承諾なしに，書籍，雑誌，Webといった媒体の形態を問わず，転載，複写することを禁じます．

▶コピーサービスのご案内

本誌バックナンバーの掲載記事については，在庫(原則として24か月分)のないものに限りコピーサービスを行っています．コピー体裁は雑誌見開きの，複写機による白黒コピーです．なお，コピーの発送には多少時間がかかる場合があります．

- コピー料金(税込み)
  1ページにつき100円
- 発送手数料(判型に関わらず)
  1～10ページ：100円，11～30ページ：200円，31～50ページ：300円，51～100ページ：400円，101ページ以上：600円
- 送付金額の算出方法
  総ページ数×100円＋発送手数料
- 入金方法
  現金書留か郵便小為替による郵送
- 明記事項
  雑誌名，年月号，記事タイトル，開始ページ，総ページ数
- 宛て先
  〒170-8461　東京都豊島区巣鴨1-14-2
  CQ出版株式会社　コピーサービス係
  (TEL：03-5395-4211，FAX：03-5395-1642)

▶お問い合わせ先のご案内

- 在庫，バックナンバー，年間購読送付先変更に関して
  販売部：03-5395-2141
- 広告に関して
  広告部：03-5395-2133
- 雑誌本文に関して
  編集部：03-5395-2122

記事内容に関するご質問は，返信用封筒を同封して編集部宛てに郵送してくださるようお願いいたします．筆者に回送してお答えいたします．

Interface

©CQ出版(株)　2003　振替　00100-7-10665
2003年6月号　第29巻　第6号(通巻第312号)
2003年6月1日発行(毎月1日発行)
定価は裏表紙に表示してあります

発行人／蒲生良治
編集人／相原　洋
編集／大野典宏　村上真紀　山口光樹　小林由美子
デザイン・DTP／クニメディア株式会社
表紙デザイン／株式会社プランニング・ロケッツ
本文イラスト／森　祐子
広告／澤辺　彰　中元正夫　渡部真美

発行所／CQ出版株式会社　〒170-8461　東京都豊島区巣鴨1-14-2
電話／編集部(03)5395-2122　URL http://www.cqpub.co.jp/interface/
広告部(03)5395-2133　インターフェース編集部へのメール
販売部(03)5395-2141　supportinter@cqpub.co.jp
CQ Publishing Co.,Ltd.／1-14-2 Sugamo, Toshima-ku, Tokyo 170-8461, Japan
印刷／クニメディア株式会社　美和印刷株式会社
製本／星野製本株式会社

日本ABC協会加盟誌
(新聞雑誌部数公査機構)　ISSN0387-9569

Printed in Japan

本誌は有名書店ほか，コンビニエンス・ストア「セブン-イレブン」でも予約購読ができます．詳しくは，お店にお問い合わせください．